

Timothy W. Simpson  
Jianxin (Roger) Jiao  
Zahed Siddique  
Katja Hölttä-Otto *Editors*

# Advances in Product Family and Product Platform Design

Methods & Applications



Springer

# Advances in Product Family and Product Platform Design



Timothy W. Simpson • Jianxin (Roger) Jiao  
Zahed Siddique • Katja Hölttä-Otto  
Editors

# Advances in Product Family and Product Platform Design

Methods & Applications

 Springer



*Editors*

Timothy W. Simpson  
Department of Mechanical  
and Nuclear Engineering  
Pennsylvania State University  
University Park, PA, USA

Jianxin (Roger) Jiao  
School of Mechanical Engineering  
Georgia Institute of Technology  
Atlanta, GA, USA

Zahed Siddique  
School of Aerospace  
and Mechanical Engineering  
University of Oklahoma  
Norman, OK, USA

Katja Hölttä-Otto  
Engineering Product Development  
Singapore University of Technology  
and Design  
Singapore, Singapore

ISBN 978-1-4614-7936-9

ISBN 978-1-4614-7937-6 (eBook)

DOI 10.1007/978-1-4614-7937-6

Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2013945755

© Springer Science+Business Media New York 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# Preface

Increased product variety, decreased costs, faster time to market. The motives for designing product platforms and developing families of products have changed little these past three decades; however, never have companies had more imperative to pursue platform-based product development. The rise of the BRICs (Brazil, Russia, India, and China), the Great Recession, and the interconnected global economy are but a few of the many factors that are causing renewed and continued interest in product platforms and product family design. In our own teaching, we have observed this shift as well. Shortly after our first book came out, industry interest was primarily focused on what is a platform and what are its potential benefits to a company, and only a few companies were aggressively pursuing platform-based product development strategies. Now, particularly in the last 2–3 years, industry interest has noticeably shifted to the implementation and execution of platforms (e.g., how do we design platforms? what constitutes a good platform? how does our platform compare to what our competitors are doing?), and we find the majority of companies are investing significant time and resources to develop a product platform and corresponding product family.

So what is a product family? Most generally, it is a set of products that share one or more common “elements” (e.g., components, modules, subsystems, fabrication processes, assembly operations) yet target a variety of different market segments. The commonality in the family is intentional—not coincidental—and arises from the product platform around which the family is derived. The individual product variants can be derived from the platform by adding, substituting, or subtracting one or more modules from the family to create a module-based product family or by “scaling” or “stretching” the platform in one or more dimensions to realize a scale-based product family (Jiao et al. 2007). Of course, it is never that straightforward in practice, as different product families require different combinations of modularity and scaling to achieve sufficient variety for the marketplace while remaining cost-effective and competitive.

Making the case for platforming in a company remains a challenge. It requires a different mindset than one for a single product, and most companies are not prepared to think across multiple generations of products and long term about

their product lines. The concept of a “market attack plan” remains foreign to many companies as they still have a single product mindset and overcoming the corporate inertia to change that takes time and energy—and lots of it. A successful platform is as much about the technical solution as it is about the financial benefits or the organizational roadmap needed to establish and follow through on a viable platform strategy. The traditional thinking and established practices to manage *product* development often do not readily translate to *platform* development—a company cannot simply substitute one word for another in an org chart or a gate review process and expect things to go smoothly.

Cross-functional product development teams, support from upper management, platform architecting, understanding the market, and financial planning are just as important now as they were when we analyzed industry trends seven years ago (Simpson et al. 2006). We have also seen that platforming in “nontraditional” areas (e.g., software, services) continues to grow and thinking globally about platforms has become the rule not the exception as companies seek to establish a presence in multiple markets around the world. The variability that this creates—in customer needs, regulations/standards, and the general business environment—can be overwhelming, and companies need to think seriously about what platform strategy is best for them, if any. In some cases, the added cost and complexity of platform-based product development may lead to undesirable products; however, careful planning and an honest assessment of the true benefits of platforming within a company often yield exciting results.

To help companies with their platform journey—and it truly is a journey that does not happen overnight—we present *Advances in Product Family and Product Platform Design: Methods and Applications*, a follow-up to our first edition, which is now 8 years old (Simpson et al. 2005). While the methods and tools from our first edition are still readily applicable, numerous advances have been made, and the applications are becoming dated and no longer reflect the variety of areas that are now being targeted by platforms (e.g., software, services). Chapter 1 in the present text reviews recent literature to bring the reader up to speed on the recent developments. The remainder of the book is organized into four parts based on the order of a typical platform development life cycle:

- Part I: Platform Planning and Strategy
- Part II: Platform Architecting and Design
- Part III: Product Family Development and Implementation
- Part IV: Applications and Case Studies

Highlights of the chapters in each part follow.

## **Part I: Platform Planning and Strategy**

The first part of the book provides a collection of methods and tools to help plan the platform development with given benefits in mind. Chapter 2 explores the benefits and pitfalls of commonality and provides evidence from several in-depth case

studies on the cost savings and commonality premiums that companies were able to achieve in a range of industries. Chapter 3 investigates the challenges of integrating customer diversity across multiple market segments and provides methods to coevolve market segments and product variants to realize novel product platforms for multiple domains. Chapter 4 provides an overview of Modular Functional Deployment, a popular method in industry to support module-based product family design and examines the impact of different module drivers on both product and platform architecting. Chapter 5 expands on the notion of parts reuse to the reuse of design information and other generic assets to leverage platforms to integrate product and production systems. Chapter 6 introduces data mining techniques to help designers quantify the relevance (or obsolescence) of product features when developing a platform and corresponding family of products. Finally, Chap. 7 discusses platform valuation tools and the use of options to support module development decisions in uncertain market environments.

## **Part II: Platform Architecting and Design**

The second part consists of eight chapters that introduce methods to help architect the platform, including methods for architecture decomposition as well as for both scalable and modular product platforms. Chapter 8 introduces a method to proactively create a platform based on assessment of market needs followed by identification of modules for individual product variants. Chapter 9 investigates the role of architecture decomposition and the impact that granularity has on modularity. Chapter 10 provides a comprehensive toolkit to support modular platform development along with an industry example to demonstrate its application. Chapter 11 explores the challenges of simultaneously designing a product platform and a product family and offers computational tools to optimize both at the same time. Chapter 12 provides a one-step approach to identify the platform and design the family of products simultaneously. Meanwhile, Chap. 13 identifies a tool chain to link disparate methods together to support product platform architecting. Chapter 14 describes a method for scale-based product family design using Quality Function Deployment (QFD) to optimize the engineering characteristics of the platform and the individual product variants. Finally, Chap. 15 offers a multi-platform approach to balance the trade-off between commonality and individual product performance that lies at the heart of product family design.

## **Part III: Product Family Development and Implementation**

The third part continues to introduce methods for product platform development but with an emphasis on the implementation and execution of the platform strategy. Chapter 16 introduces methods and tools to support global platform design that integrates modularity and supply chain decisions. Chapter 17 presents three tools to support system architecting by linking functions, behaviors, and working principles

to a variety of customer requirements. Chapter 18 discusses three methods to help identify potential common components in a product family and visualize the respective performance trade-offs. Chapter 19 describes several commonality indices and investigates their ability to capture the total cost savings within the product family. Chapter 20 investigates the implications of managing multiple design projects during product family development and introduces a process architecture to support modular design project planning. Chapter 21 discusses the challenges when architecting software platforms and codifies design principles to support software reuse. Chapter 22 explores the influences and impact of human variability on product design and identifies basic scenarios where platforming and modularity are advantageous. Finally, Chap. 23 concludes this part with a series of recommendations to align the product family with the manufacturing and supply chain while stressing the importance of aligning market variety with design versatility and supply chain responsiveness.

## Part IV: Applications and Case Studies

The fourth part provides a series of practical examples from industry. In Chap. 24, a modular architecture is developed for a cordless handheld vacuum cleaner using Modular Function Deployment, which was introduced in Chap. 4. Chapter 25 investigates opportunities for commonality between different classes of ships for the US Coast Guard. Chapter 26 discusses heuristics for architecting software-intensive families, which are then used to develop a software platform for a family of industrial machines. Chapter 27 uses a sequence of design tools discussed in the book to analyze customer requirements and subsequently design a family of electric violins. Chapter 28 examines the implications of product family design and reuse on product life cycles with a smartphone case study. Chapter 29 describes the application of the Generational Variety Index (Martin and Ishii 2002) to analyze four generations of Apple's iPhone product line. A family of leaf blowers is designed using the proactive modular platform design method introduced in Chap. 8. Finally, the book concludes with an Epilogue that offers future research directions and discusses several trends shaping future applications of product platform and product family design and development.

University Park, PA, USA  
Atlanta, GA, USA  
Norman, OK, USA  
Singapore

Timothy W. Simpson  
Jianxin (Roger) Jiao  
Zahed Siddique  
Katja Hölttä-Otto

## References

- Jiao RJ, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18(1):5–29
- Martin MV, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. *Res Eng Des* 13(4):213–235
- Simpson TW, Marion TJ, de Weck O, Holtta-Otto K, Kokkolaras M, Shooter SB (2006) Platform-based design and development: current trends and needs in industry. In: ASME design engineering technical conferences – design automation conference ASME, Philadelphia, Pennsylvania, Paper No. DETC2006/DAC-99229
- Simpson TW, Siddique Z, Jiao J (2005) Product platform and product family design: methods and applications. Springer, New York, NY



# Contents

<b>1</b>	<b>A Review of Recent Literature in Product Family Design and Platform-Based Product Development . . . . .</b>	<b>1</b>
	Zhila Pirmoradi, G. Gary Wang, and Timothy W. Simpson	
<b>Part I Platform Planning and Strategy</b>		
<b>2</b>	<b>Crafting Platform Strategy Based on Anticipated Benefits and Costs . . . . .</b>	<b>49</b>
	Bruce G. Cameron and Edward F. Crawley	
<b>3</b>	<b>Multidisciplinary Domains Association in Product Family Design . . . . .</b>	<b>71</b>
	Hoda ElMaraghy and Tarek AlGeddawy	
<b>4</b>	<b>Modular Function Deployment: Using Module Drivers to Impart Strategies to a Product Architecture . . . . .</b>	<b>91</b>
	Mark W. Lange and Andrea Imsdahl	
<b>5</b>	<b>Emphasizing Reuse of Generic Assets Through Integrated Product and Production System Development Platforms . . . . .</b>	<b>119</b>
	Hans Johannesson	
<b>6</b>	<b>Quantifying the Relevance of Product Feature Classification in Product Family Design . . . . .</b>	<b>147</b>
	Conrad S. Tucker	
<b>7</b>	<b>Platform Valuation for Product Family Design . . . . .</b>	<b>179</b>
	Seung Ki Moon and Timothy W. Simpson	



## Part II Platform Architecting and Design

- 8 A Proactive Scaling Platform Design Method  
Using Modularity for Product Variations . . . . .** 201  
Keith Hirshburg and Zahed Siddique
- 9 Architectural Decomposition: The Role of Granularity  
and Decomposition Viewpoint . . . . .** 221  
Katja Hölttä-Otto, Noemi Chiriac, Dusan Lysy, and Eun Suk Suh
- 10 Integrated Development of Modular Product Families:  
A Methods Toolkit . . . . .** 245  
Dieter Krause, Gregor Beckmann, Sandra Eilmus,  
Nicolas Gebhardt, Henry Jonas, and Robin Rettberg
- 11 Solving the Joint Product Platform Selection and Product  
Family Design Problem: An Efficient Decomposed  
Multiobjective Genetic Algorithm with  
Generalized Commonality . . . . .** 271  
Aida Khajavirad, Jeremy J. Michalek, and Timothy W. Simpson
- 12 One-Step Continuous Product Platform Planning:  
Methods and Applications . . . . .** 295  
Achille Messac, Souma Chowdhury, and Ritesh Khire
- 13 Defining Modules for Platforms: An Overview  
of the Architecting Process . . . . .** 323  
Katja Hölttä-Otto, Kevin N. Otto, and Timothy W. Simpson
- 14 A QFD-Based Optimization Method for Scalable  
Product Platform . . . . .** 343  
Xinggang Luo, Jiafu Tang, and C.K. Kwong
- 15 Cascading Platforms for Product Family Design . . . . .** 367  
Jiju A. Ninan and Zahed Siddique

## Part III Product Family Development and Implementation

- 16 Global Product Family Design: Simultaneous Optimal  
Design of Module Commonalization and Supply Chain  
Configuration . . . . .** 393  
Kikuo Fujita
- 17 Architecture-Centric Design Approach  
for Multidisciplinary Product Development . . . . .** 419  
A.A. Alvarez Cabrera, H. Komoto, T.J. van Beek,  
and T. Tomiyama

**18 Product Family Commonality Selection Using Optimization and Interactive Visualization . . . . . 449**  
 Ritesh Khire, Jiachuan Wang, Trevor Bailey, Yao Lin, and Timothy W. Simpson

**19 Developing and Assessing Commonality Metrics for Product Families . . . . . 473**  
 Michael D. Johnson and Randolph E. Kirchain

**20 Managing Design Processes of Product Families by Modularization and Simulation . . . . . 503**  
 Qianli Xu and Roger J. Jiao

**21 Design Principles for Reusable Software Product Platforms . . . . . 533**  
 Carlos O. Morales

**22 Considering Human Variability When Implementing Product Platforms . . . . . 559**  
 Christopher J. Garneau, Gopal Nadadur, and Matthew B. Parkinson

**Part IV Applications and Case Studies**

**23 Building, Supplying, and Designing Product Families . . . . . 589**  
 David M. Anderson

**24 Modular Function Deployment Applied to a Cordless Handheld Vacuum . . . . . 605**  
 Fredrik Börjesson

**25 Optimal Commonality Decisions in Multiple Ship Classes . . . . . 625**  
 Michael J. Corl, Michael G. Parsons, and Michael Kokkolaras

**26 A Heuristic Approach to Architectural Design of Software-Intensive Product Platforms . . . . . 647**  
 Carlos O. Morales

**27 Customer Needs Based Product Family Sizing Design: The Viper Case Study . . . . . 683**  
 Cassandra Sotos, Gül E. Okudan Kremer, and Gülşen Akman

**28 Product Family Design and Recovery for Lifecycle . . . . . 707**  
 Minjung Kwak and Harrison Kim

**29 Application of the Generational Variety Index: A Retrospective Study of iPhone Evolution . . . . . 737**  
 Gopal Nadadur, Matthew B. Parkinson, and Timothy W. Simpson

**30 Designing a Lawn and Landscape Blower Family  
Using Proactive Platform Design Approach . . . . . 753**  
Keith Hirshburg and Zahed Siddique

**Epilogue . . . . . 777**  
Timothy W. Simpson, Roger J. Jiao, Zahed Siddique,  
and Katja Hölttä-Otto

**References . . . . . 789**

**Index . . . . . 793**

# Contributors

**Gülşen Akman** Kocaeli University, Kocaeli, Turkey

**Tarek AlGeddawy** Intelligent Manufacturing (IMS) Centre, University of Windsor, ON, Canada

**David M. Anderson** Build-to-Order Consulting, Cambria, CA, USA

**Trevor Bailey** United Technologies Research Center, East Hartford, CT, USA

**Gregor Beckmann** Institute for Product Development and Mechanical Engineering Design, Hamburg University of Technology, Hamburg, Germany

**Fredrik Börjesson** Modular Management USA, Inc., Bloomington, MN, USA

**A.A. Alvarez Cabrera** Delft University of Technology, Delft, The Netherlands

**Bruce G. Cameron** Massachusetts Institute of Technology, Cambridge, MA, USA

**Noemi Chiriac** Department of Mechanical Engineering, University of Massachusetts Dartmouth, North Dartmouth, MA, USA

**Souma Chowdhury** Department of Mechanical and Aerospace Engineering, Syracuse University, Syracuse, NY, USA

**Michael J. Corl** CDR, U.S.C.G., U.S. Coast Guard Academy, New London, CT, USA

**Edward F. Crawley** Massachusetts Institute of Technology, Cambridge, MA, USA

**Sandra Eilmus** Institute for Product Development and Mechanical Engineering Design, Hamburg University of Technology, Hamburg, Germany

**Hoda ElMaraghy** Intelligent Manufacturing (IMS) Centre, University of Windsor, ON, Canada

**Kikuo Fujita** Department of Mechanical Engineering, Graduate School of Engineering, Osaka University, Osaka, Japan

**Christopher J. Garneau** OPEN Design Lab, The Pennsylvania State University, University Park, PA, USA

U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, USA

**Nicolas Gebhardt** Institute for Product Development and Mechanical Engineering Design, Hamburg University of Technology, Hamburg, Germany

**Keith Hirshburg** School of Aerospace and Mechanical Engineering, University of Oklahoma, Norman, OK, USA

**Katja Hölttä-Otto** Engineering Product Development, Singapore University of Technology and Design, Singapore

**Andrea Imsdahl** Department of Applied Mechanical Engineering, Industrial Engineering and Management, Royal Institute of Technology, Södertälje, Sweden

**Roger J. Jiao** Georgia Institute of Technology, Woodruff School of Mechanical Engineering, Atlanta, GA, USA

**Hans Johannesson** Department of Product and Production Development, Chalmers University of Technology, Gothenburg, Sweden

**Michael D. Johnson** Department of Engineering Technology and Industrial Distribution, Texas A&M University, College Station, TX, USA

**Henry Jonas** Institute for Product Development and Mechanical Engineering Design, Hamburg University of Technology, Hamburg, Germany

**Aida Khajavirad** Carnegie Mellon University, Pittsburgh, PA, USA

**Ritesh Khire** United Technologies Research Center, East Hartford, CT, USA

**Harrison Kim** Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA

**Randolph E. Kirchain** Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, USA

**Michael Kokkolaras** McGill University, Montreal, QC, Canada

**H. Komoto** National Institute of Advanced Industrial Science and Technology, Tsukuba, Ibaraki, Japan

**Dieter Krause** Institute for Product Development and Mechanical Engineering Design, Hamburg University of Technology, Hamburg, Germany

**Gül E. Okudan Kremer** The Pennsylvania State University, University Park, PA, USA

**Minjung Kwak** Department of Industrial and Information Systems Engineering, Soongsil University, Seoul, Korea

**C.K. Kwong** Department of Industrial and System Engineering, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

**Mark W. Lange** Department of Applied Mechanical Engineering, Industrial Engineering and Management, Royal Institute of Technology, Södertälje, Sweden

**Yao Lin** United Technologies Research Center, East Hartford, CT, USA

**Xinggang Luo** State Key Lab of Synthetic Automation of Process Industries, School of Information Science and Engineering, Northeastern University, Shenyang, Liaoning, People's Republic of China

**Dusan Lysy** Xerox Corporation, Webster, NY, USA

**Achille Messac** Bagley College of Engineering, Mississippi State University, Mississippi State, MS, USA

**Jeremy J. Michalek** Carnegie Mellon University, Pittsburgh, PA, USA

**Seung Ki Moon** School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore

**Carlos O. Morales** Animas Corporation, Johnson & Johnson Medical Devices, West Chester, PA, USA

**Gopal Nadadur** OPEN Design Lab, The Pennsylvania State University, University Park, PA, USA

**Jiju A. Ninan** Schlumberger, Sugarland, TX, USA

**Kevin N. Otto** Singapore University of Technology and Design, Singapore

**Matthew B. Parkinson** OPEN Design Lab Engineering Design, Mechanical Engineering, and Industrial Engineering, The Pennsylvania State University, University Park, PA, USA

**Michael G. Parsons** University of Michigan, Ann Arbor, MI, USA

**Zhila Pirmoradi** Product Design and Optimization Laboratory, School of Mechatronic Systems Engineering, Simon Fraser University, Burnaby, BC, Canada

**Robin Rettberg** Institute for Product Development and Mechanical Engineering Design, Hamburg University of Technology, Hamburg, Germany

**Zahed Siddique** School of Aerospace and Mechanical Engineering, University of Oklahoma, Norman, OK, USA

**Timothy W. Simpson** Mechanical and Nuclear Engineering, Penn State University, University Park, PA, USA

Industrial and Manufacturing Engineering, Penn State University, University Park, PA, USA

**Cassandra Sotos** The Pennsylvania State University, University Park, PA, USA

**Eun Suk Suh** Department of Industrial Engineering, Seoul National University, Seoul, South Korea

**Jiafu Tang** State Key Lab of Synthetic Automation of Process Industries, School of Information Science and Engineering, Northeastern University, Shenyang, Liaoning, People's Republic of China

**T. Tomiyama** Cranfield University, Cranfield, UK

**Conrad S. Tucker** Industrial Engineering and Engineering Design, The Pennsylvania State University, University Park, PA, USA

**T.J. van Beek** Delft University of Technology, Delft, The Netherlands

**G. Gary Wang** Product Design and Optimization Laboratory, School of Mechatronic Systems Engineering, Simon Fraser University, Burnaby, BC, Canada

**Jiachuan Wang** United Technologies Research Center, East Hartford, CT, USA

**Qianli Xu** Agency for Science, Technology and Research, Institute for Infocomm Research, Singapore

# Chapter 1

## A Review of Recent Literature in Product Family Design and Platform-Based Product Development

Zhila Pirmoradi, G. Gary Wang, and Timothy W. Simpson

**Abstract** Increased demand for a greater variety of consumer products has forced many companies to rethink their strategies to offer more product variants. For manufacturers, producing a variety of products can satisfy this increasing demand and help companies gain more of market share; however, increased variety can lead to higher design and production costs as well as longer lead times for new variants. As a result, a trade-off arises between cost-effectiveness and satisfying diverse customer demand. Research has found that such a trade-off can be properly managed by exploiting product family design (PFD) and platform-based product development, an area that has been widely studied for the past two decades. New approaches have been proposed to address different issues related to PFD and platform development. Performance of these approaches has been assessed through case studies and applications to different industry sectors. This chapter focuses on reviewing the research in this field to classify recent advancements in PFD and platform development. We identify new achievements with regard to multiple aspects of PFD: customer involvement in design, market-driven studies, metrics for assessing platforms and families, indices for platform and family design, product family optimization issues, platform development issues, and, finally, issues relevant to supporting future platform design. Through a comparison with previous research studies, we identify ongoing challenges in this field along with potential directions for new research.

---

Z. Pirmoradi • G.G. Wang (✉)

Product Design and Optimization Laboratory, School of Mechatronic Systems Engineering,  
Simon Fraser University, Burnaby, BC, Canada  
e-mail: [gary\\_wang@sfu.ca](mailto:gary_wang@sfu.ca)

T.W. Simpson

Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA  
Industrial and Manufacturing Engineering, Penn State University, University Park,  
PA 16802, USA



## 1.1 Introduction

Customers' needs continually evolve and shift over time, and their demand for product variety and newer versions of products has increased rapidly in recent decades. Many companies have been attempting to provide more product variants to satisfy the increasing demand of niche segments in the market without sacrificing production efficiency (Berry and Pakes 2007; Jiao et al. 2007c). A trade-off quickly emerges: satisfying this wide array of customer needs leads to more sales, but producing this variety of products often increases costs and makes companies less profitable. In other words, using more common features and components in production (i.e., standardization of components) can reduce market share if products are not sufficiently differentiated. One way to manage this confliction is *mass customization*, which enables economies of scale and satisfaction of diverse expectations concurrently (Jiao and Zhang 2005). Mass customization emerged in the early 1990s with the objective of satisfying individual customers through increased product variety (Pine 1993). Product family design and platform-based product development are effective strategies to enable mass customization as they can effectively provide variety at reduced costs (Marion et al. 2007; Park and Simpson 2008). A product family can be considered as a set of products that share a number of common components and functions with each product having its unique specifications to meet demands of certain customers (Meyer and Lehnerd 1997). The common parts are usually defined as the product platform (Simpson et al. 2005). Design and development of families of products is challenging for many aspects. It involves selecting business strategies, considering multiple marketing issues, engineering customer needs, studying customer behavior and choice-related issues, as well as carefully considering engineering aspect of design, such as manufacturability, technological aspects, and design support issues (i.e., modeling and developing design information depository).

In general, the product family development process can be divided into three stages. The first stage is to translate identified customer needs (CN) into functional requirements (FR) for a product. The second stage deals with mapping those requirements into proper design variables (DV) subject to potential manufacturing constraints. The third stage is the process planning and determining process variables, which will be followed by the supply chain platform design, and determination of proper values for logistics variables (LV) (Jiao et al. 2007c). Figure 1.1 provides an illustration of the aforementioned issues and concepts.

According to this figure, an outline of this study can be provided as follows. As the front-end issues include customer involvement, product portfolio design, product family positioning, and transition or mapping from customer needs to functional requirements, these aspects will be discussed in the first section of this study. The product family design issues include the product family configuration, product architecture, design of families and platforms, variety in design, leveraging commonality and modularity, optimization of the family and platform design, and design decision support systems. The second section in the study will address

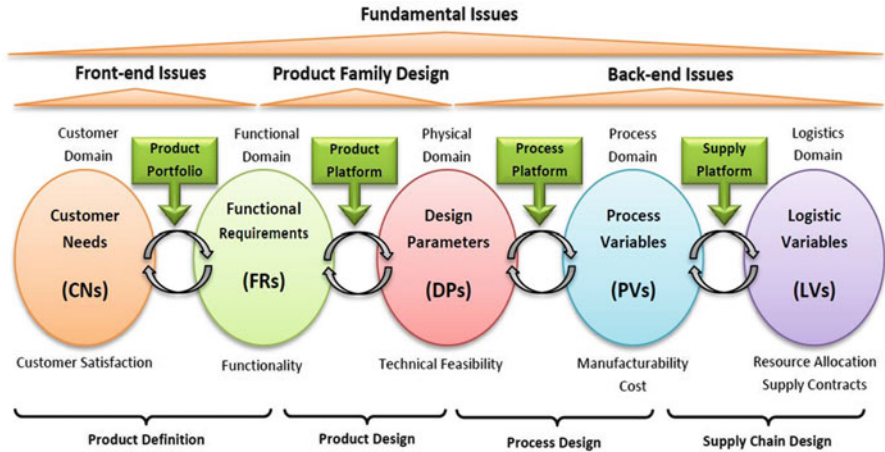


Fig. 1.1 A general view of product family design and development (Jiao et al. 2007c)

concerns directly related to design. Finally, the back-end issues will be discussed in the third section, including manufacturability, cost considerations, supply chain design and management, metrics for product family and platform design, resource allocation, redesign and flexible platform design issues, processes design, and process platforms. Eventually, the areas of improvement and potential areas for future research, as well as summary remarks, will be presented at the end of this study.

Researchers have developed techniques to address these stages individually and as a whole, and they have assessed efficiency of their developments through case studies. Among the developments with regard to PFD, two are well known: the top-down and the bottom-up approaches. The top-down approach focuses on developing a family in order to fulfill a variety of customer needs. The bottom-up approach attempts to increase production efficiency through providing solutions for reuse of components in multiple products and redesign to develop a family based on available products (Alizon et al. 2007).

## 1.2 Fundamental Concepts

The fundamental issues and concepts regarding design and development of product families are summarized in Table 1.1. For ease of reading, they are categorized into three sections, namely, the front-end, design and development, and back-end issues. A novel state-of-the-art review was implemented by Jiao et al. (2007c) for research activities before 2006. Therefore, this study focuses on published research since that time.

**Table 1.1** (a) Front-end issues, (b) design and development issues, (c) back-end issues

Concept/issue	Definition	Examples and approaches
(a)		
Product architecture	A concept for describing relations among components and connecting the functions to the components in a product. Platform architecture describes the logical relations between common and unique elements for enabling highly customized products based on customer preferences (Xu et al. 2008)	<i>Modular architecture</i> : functions-components mapping for minimizing inter-module interactions <i>Integral architecture</i> : performance-driven or cost-based architectures, enabling variety, product change, and engineering standards (Cutherell 1996)
Product family and platform configuration	Approaches for selection of platform variables versus non-platform variables and platform configuration and selection	<i>Parametric platforms</i> : finding optimal platform parameters through algorithms such as genetic algorithm <i>Configurational platforms</i> : module identification through data mining techniques, reasoning systems, clustering approaches, etc.
Product family modeling and knowledge-based systems	Approaches for knowledge integration about product families and platforms	Unified modeling approach, graph grammar approach, architecture modeling, set-based models, parametric modeling, functional models, configuration information modeling, etc.
Product portfolio positioning	Strategies to give the customers the exact number of variants that they need	Factor analysis, discriminant analysis, perception scaling techniques, choice set determination, discrete choice analysis, probabilistic choice modeling
(b)		
Platform	Sets of components, technologies, subsystems, processes, and interfaces that form a structure to develop a number of products to maximize commonality and minimize individual performance deviations (Li et al. 2007)  Product platform design deals with determination of the variables to be shared, as well as optimal values for both shared and unique variables among variants	<i>Scalable platforms</i> : variants can be produced through shrinkage or extension of scalable variables <i>Modular platforms</i> : enabling product differentiation through adding/removing/substituting different modules <i>Generational platforms</i> : enabling consideration of possible requirements for changing the design over a period of time, to allow variation of next generations (Jiao et al. 2007c)
Variety versus commonality	The diversity that a production system can provide for the market. Two main concepts are	Functional variation: driven by customer requirements and it

(continued)

**Table 1.1** (continued)

Concept/issue	Definition	Examples and approaches
	modularity and commonality. Modularity decomposes components and functions into independent groups, while commonality clusters the components and functions based on similarity or other criteria	attempts to increase variation in functions Technical variation: addresses issues such as manufacturability and costs and it tries to decrease diversity in functions and processes
	Functional variation can be traced in research studies focusing on product line structuring, product portfolio positioning, product pricing, and portfolio optimization. Technical variation strategy, however, can be found in areas such as design for variety, variety reduction schemes, and modularization (Jiao et al. 2007b)	Functional commonality/modularity: paying attention to the relation between customer needs and functional requirements Technical commonality/modularity: based on technical design solutions or physical, based on mapping of functional requirements into design parameters (Jiao et al. 2007b)
Design optimization	Techniques and algorithms for determining the optimal design variable values, for specific objectives subject to constraints	Search approaches, multi-attribute utility analysis, preference-based design, MDO methods, product line design through cost modeling, engineering design considerations, etc.
Design support systems	Tools for facilitating information management and creating design information repository	Configuration reasoning systems, agent-based knowledge management systems, knowledge-intensive evaluation models, advisory systems, web-based customization systems, etc.
(c)		
Manufacturability	All concerns related to manufacturing part of PFD, including standardization and commonalization of the processes, facilities, and technologies and controlling manufacturing costs and process variations that result from customization development risks reduction techniques	Exploiting process families/platforms Developing generic bill of materials (GBOMs) Integrating mass customization with product life cycle management Design for reuse Flexible platform design Reconfigurable system design (Mehrabi et al. 2000)
Metrics and indices	Tools that provide information about the cost savings resulted from commonalization compared to quantitative benefits of commonality	CMC, DCI, TCCI, PCI, %C, TCM, etc. (see Sect. 1.4.1 for more detail)
Supply chain management	Consideration of issues related to different PFD stages, including	Developing robust approaches to maximize profit of the supply

(continued)

**Table 1.1** (continued)

Concept/issue	Definition	Examples and approaches
	first stages of supply chain management to the other end, which is delivery to customers	chain, along with minimizing delivery time, procurement costs, logistic and assembly costs, etc.
Postponement	Enabling late differentiation and allowing reduced inventory and delaying the resources commitment to the final configuration of a product as long as possible	Redesigning the family architecture or rescheduling the master production plan can facilitate postponement

While many approaches have been developed in the past decade to address PFD and platform development issues, many challenges remain. Necessity of handling many variables and simultaneous consideration of interdependencies among different elements of design make PFD problems complex in nature. Research continues in this field, and many opportunities exist for improvement. Based on the categories and concepts introduced in this section, new developments and recent achievements of this area are reviewed in this chapter. The literature review of each issue is presented in subsequent sections, and identified opportunities for future research are addressed in the closing section.

### 1.3 Front-End Issues in Product Family Design

As mentioned in the introduction, in the front-end of product family development, the following issues are of interest: involving customers into product characterization, product family positioning and market segmentation, product portfolio design, realizing the customer needs and the required functions to meet those needs, and eventually tools or techniques for facilitating information standardization and leveraging the product family knowledge.

#### 1.3.1 *Product Portfolio and Product Family Positioning*

While different variants in a family may call for similar technical and manufacturing requirements, research has shown that they might not be equally preferred by the customers (Thevenot and Simpson 2007b). Therefore, product family positioning is vital for properly balancing the diverse customer tastes and manufacturing costs of variation (Olewnik and Lewis 2006). The product family positioning problem is a front-end issue that can be facilitated through market segmentation that segments the market into different clusters, providing specific

variants for each, and identifying opportunities for adjusting products in order to attract more customers (Hisrich and Peters 1991). The family positioning problem focuses on increasing variation and diversity in the offered products. Clustering is required for this target so that the minimum possible number of variants which cover the maximum possible customer preferences can be determined. As a result, attention has been paid to clustering techniques such as data mining, fuzzy clustering, conjoint analysis, and heuristic search algorithms for finding matches between customer groups and product variants.

The summary of research done in area of positioning is as follows:

- Review of fuzzy logic applications for product family development (Barajas and Agard 2009).
- Use of engineering characteristics as segmentation variables for market segmentation and product line positioning (Zhang et al. 2007).
- Use of conjoint analysis for identifying customer needs and clustering method for segmenting customers (Kazemzadeh et al. 2009).
- Development of three indices: cost reduction, commonality percentage, and satisfaction percentage for comparing a generic product for the whole market with a customized product for each segment.
- Consideration of different market leveraging strategies and product line extension (Doraszelski and Draganska 2006).
- *Product mix selection* (Chung et al. 2008) assuming varying demand and different priorities for customer orders.
- Use of *real options* (Jiao et al. 2006; Jiao 2012) to assess values of different design options for flexible and intime action against market volatilities.

Sharman and Yassine (2007) showed that serious difficulties can arise when using the real options for clustering, especially when the product architecture is complex and it lacks a truly hierarchical structure. However, market conditions play an essential role in product family and platform design decisions, and many other factors affect the market, which affect planning and decision-making in turn. For example, markets are affected by government policies, demographics, and personal characteristics of customers as well as customer purchasing behaviors and preferences, which may vary under different circumstances. Therefore, prior assessment of all such issues is necessary for making the most reliable decisions before development of any product family. This makes PFD even more challenging, as it exacerbates development time and information gathering. Meanwhile, value creation for customers and successful development of product families/platforms depend on agile action for capturing such changing trends. Therefore, the demand for further research and more scrutiny in this filed continues to grow.

### 1.3.2 Market-Driven Product Family Design

The involvement of customer preferences into engineering design decisions has received remarkable attention recently. Mapping the customer needs into functional requirements of products, mapping the customer requirements into different market segments (Farrell and Simpson 2008), leveraging tools such as choice modeling for predicting customer reactions in different situations, using *quality function deployment (QFD)* for translating customer requirements into design requirements (Li et al. 2006), and other similar techniques have been helpful in product attributes selection, family/platform configuration, and portfolio optimization to fulfill the market demands.

A market-driven product family design approach was proposed by Kumar et al. (2006), known as *MPFD* to integrate market considerations with family design concerns in order to enable product family positioning. This approach examines the impact of variety on different market niches and employs a demand modeling through which the impacts of competition in different segments on market share of each competition can be identified. Similarly, in another study the same researchers integrated market share considerations and demand modeling for simultaneously optimizing decisions related to platform leveraging and product line positioning (Kumar et al. 2009). Their purpose was to overcome limitations of past market-driven PFD studies, which have only considered production line positioning in lieu of how a product competes with other products by the same supplier as well as competitive products. They concluded that including performance considerations and market considerations results in obtaining more economic decisions.

The list of other works done with consideration of market is as follows:

- Providing suggestions for overcoming the shortcomings of previous product line optimization models (Michalek et al. 2011).
- Exploring the strengths and limitations of discrete choice models for understanding market demand and supporting mass customization (Ferguson et al. 2011).
- Using the hierarchical mixed logit model for continuous representation of preference heterogeneities and the latent multinomial logit model for the discrete representation (Sullivan et al. 2011).
- Simultaneously considering important factors from both marketing and engineering domains (Luo 2011) considering the strategic reactions of incumbent manufacturers and retailers.

Among the works done for optimizing the product portfolio, few take the advantage of customer-perceived value; the work (Farrell and Simpson 2008) is an example that focuses on the set of components with the highest potential for cost savings, rather than redesigning the entire product line. More development potential for preserving customer-perceived variety exists, as it will help in offering the optimum number of products that is not necessarily equal to the number of market segments as the results of the study (Michalek et al. 2011) show.

Also, most of the studies that have included market systems into product development apply only to a single product, and there are numerous opportunities to cover in product family design. For example, Shiau and Michalek (2009b) studied two factors that affect design stemming from market systems: (1) the interaction structure between manufacturer and retailer and (2) the heterogeneity of consumer preferences in modeling. Such considerations can be applied to product family design as well.

### 1.3.3 Product Family Modeling

Modeling product families and platforms can serve as a basis for prior analysis of design, and it can play a significant role in the early stages of product family design and platform development prior to any redesigns or design implementations. The purpose of studies about family modeling can be identification of modules and unique components as Zhang et al. (2006c) have proposed to decrease the effect of module's internal behaviors on external interactions among them to decrease complexity of the modularized design. The applied approaches and their objectives can be summarized as follows.

An important yet not widely addressed issue in PFD is the end-of-life assessments for families. As the variants in a family will be taken out of the production cycle after their life, there might be plenty of opportunities for taking advantage of these designs and enabling economic recovery of the design for new generations. One study tackling this issue is Kwak and Kim (2011), which assesses the family design through a quantitative model from this perspective. Also, integration of life cycle management issues into mass customization (Zhang and Fan 2006) is to provide a multi-domain modeling of the product family architecture. There are some other important works that are listed below:

- Integration of product family data and process family data into a framework for product life cycle management (PLM) (Zhang et al. 2006b) to capture and reflect diverse relationships among model components and entities.
- Knowledge management framework development (Nanda et al. 2007) for capturing and translating the design information into a unified network, called *networked bill of material* (NBOM), for searching, reusing, aggregating, and analyzing design knowledge in a product family.
- Managing product variety and enabling efficient reuse of validated design and manufacturing data through a systematic modeling approach (Brière-Côté et al. 2010).
- Sharing of design data definitions through an extended generic product structure (Callahan 2006).
- Developing an assembly sequence model (Siddique and Wilmes 2007) to find the optimal assembly sequence which required the minimum modification of current assembly plant for adding new variants to the family based on the fact that adding a new variant needs prior considerations of cost and feasibility.



- Introducing a product family master structure (Yu and Cai 2009) to provide a basis for information reuse in product reconfiguration, using the component-based *design structure matrix* (DSM) to illustrate the hierarchical dependencies among structures and design processes.
- Developing *part relationship model* (Liu and Qi 2006; Fan and Qi 2007) for forecasting the component increase based on product proliferation through considering the evolving nature of part relations.
- Multi-agent approach for product family modeling in conceptual design (Ostrosi et al. 2011).
- Using a mathematical structure to serve as a basis for a product family algebra in features modeling (Höfner et al. 2011).

According to the literature, several attempts have been made to cover different aspects of product family design knowledge management; however, less attention has been paid to the data on the back-end stages of PFD, for example, knowledge and information about the supply chain (i.e., relations of different suppliers to efficiency of the product family and developed platform) or other examinations such as interdependencies and correlations of different stages of product family development to each other. Such issues can contribute significantly to efficiency of design solutions in the sense that they allow for more comprehensive considerations prior to decision-making and they provide the possibility of wider assessments in regard to the different stages involved in platform and family design and development.

### ***1.3.4 Platform and Product Family Configuration***

For configuration of product family and platforms, it is necessary to decide on which components or modules or features to be shared among the variants, and this requires consideration of different levels of commonality/customization. A comprehensive framework for platform planning is proposed (Chowdhury et al. 2011), which enables satisfying different market niches through a mathematical modeling in designing platforms. Their framework is called Comprehensive Product Platform Planning (CP<sup>3</sup>). Key concepts and existing approaches for configuration design such as frame-based models, case-based reasoning, variable-oriented structure configuration, and process-oriented assembly configuration are discussed in Zhao et al. (2010), which proposes an approach called *Product Family Extension Configuration Design* (PFECD). *Extension theory* is used in their study, which allows analysis of adding elements/material/relations to existing products based on existing constraints and enables managing the trade-off between mass production and individual customization.

*Appearance customization* of the products has been considered as an industrial design issue in Liu et al. (2009), which proposed a *PFD DNA* method to develop new families inheriting characteristics of existing families, while creating unique style characteristics of their own.

Since variation in the product design information can adversely affect the efficiency of design and redesign of product families, finding ways to reduce such variation can be helpful. Thevenot and Simpson (2007b) investigated sources of variation when using the product line commonality index (PCI) for estimating commonality among family members. They provide guidelines for reducing the variation resulting from product dissection. Technology changes and changing market opportunities can also impact the efficiency of the developed platforms; and Rojas and Esterman (2008) developed an *impact assessment process* and presented remedial suggestions for cases in which varying conditions after developing platforms might result in efficiency loss. In a reverse case, the application and impact of platform strategy on marketing strategies, brand, and business processes have been studied (Thomas 2012), and it has been concluded that application of platform-based planning is important for product design.

Service family design is another application derived from PFD as defined in Moon et al. (2007) on the basis of platform-based design principles. Their study used the game theory to analyze different options in module selection based on different cost strategies and under conditions with uncertain and incomplete information. This new application can be a choice for future development of service families. Service families can result in efficiency improvements in service sectors due to standardization of processes, and cost savings resulted from unification of different service activities.

## 1.4 Product Family Design and Development Issues

The middle stages that connect the back-end to the front-end in product family design include all the approaches, techniques, and developments regarding the design and realization of the chosen functionality in a family of products. In order to facilitate design of a product family, concepts such as commonality versus modularity or variety and approaches for optimization of the family/platform design are fundamental. The research improvements and developments regarding such concepts are discussed in this section.

### 1.4.1 *Commonality Versus Variety*

Leveraging commonality can lead to remarkable cost savings and higher standardization of the product line. On the other hand, variety is desired because more variation results in more customer groups' coverage and satisfying specific needs of more customers. However, variety is in conflict with commonality. Jiao et al. (2007b) considered ten outstanding papers that have presented a view of the cutting-edge research studies in the area of commonality and modularity management. Among their reviewed papers, the work by Fixson can be highlighted, as it provides a comprehensive literature review on the approaches and techniques for

**Table 1.2** Main studies including commonality indices and metrics

References	Subject	Remarks
Thevenot and Simpson (2006)	Comprehensive evaluation of the most common commonality indices	<ul style="list-style-type: none"> <li>• DCI, TCCI, PCI, %C, CI, and CI<sup>(C)</sup> were compared based on the ease of data collection, consistency, sensitivity, and their repeatability</li> </ul>
van Wie et al. (2007)	Testing the hypothesis of platform elements being more integrated and differentiating elements being more modular	<ul style="list-style-type: none"> <li>• The hypothesis could not be supported</li> <li>• Suggestion of the study is to increase commonality of platform elements and modularity of differentiating elements</li> </ul>
Thevenot and Simpson (2007a)	Introduction of a <i>comprehensive metric for commonality</i> (CMC)	<ul style="list-style-type: none"> <li>• The proposed metric can capture useful information for design and redesign of product families and resolves the trade-off between excessive commonality and excessive diversity</li> </ul>
Alizon et al. (2009a, b)	Proposal of an index that considers value increase due to diversity (CDI)	<ul style="list-style-type: none"> <li>• <i>Commonality</i> versus <i>diversity index</i> helps designers identify the components that need to be common and assigns value to desired commonality and diversity while penalizes both, if undesired</li> </ul>
Alizon et al. (2006)	Using CDI along with value analysis and <i>design structure matrix flow</i>	<ul style="list-style-type: none"> <li>• Extension of the design structure matrix (DSM) to obtain a new matrix named <math>DSM_{flow}</math>, including flow interactions between modules to improve existing families and increase customer satisfaction</li> </ul>
Blecker and Abdelkafi (2007)	Development of <i>total commonality metric</i> (TCM)	<ul style="list-style-type: none"> <li>• Enabling benchmarking to provide opportunities to redesign existing product families</li> <li>• Considering GBOM for evaluating the overall commonality within the family</li> </ul>

managing commonality and modularity in development of products and processes (Fixson 2007). Several commonality indices have been developed to provide insight and information about the cost savings of commonality, instead of direct quantification of commonality benefits (Khire et al. 2008). A commonality index can generally be defined as a metric for assessing the degree of commonality among product family members, and it can include different parameters such as the number of common components, component costs, and manufacturing processes (Thevenot and Simpson 2007a). Use of such indices is usually the first stage in designing a new product family or in analyzing an existing family (Ye et al. 2009). Table 1.2 lists the studies regarding development or application of indices for platform and family design and configuration.

Results of comparisons among these six indices and the new CMC index by Thevenot and Simpson (2007a) are summarized in Table 1.3. Taking more factors

**Table 1.3** Characteristics of six widely used commonality indices

Index	Conditions of use	Limitations	Improvement area(s)
DCI, CI <sup>(C)</sup>	<ul style="list-style-type: none"> <li>Component based (CI<sup>(C)</sup> considers process commonality too)</li> <li>Moving boundaries</li> <li>CI<sup>(C)</sup> is an extension of DCI, taking factors such as product volume and component costs into consideration</li> </ul>	<ul style="list-style-type: none"> <li>Difficult to interpret due to varying upper limit for different families</li> <li>Difficult to assess improvement of redesigns</li> <li>Not useful for large number of options</li> <li>CI<sup>(C)</sup> requires cost estimation which might be challenging</li> </ul>	<ul style="list-style-type: none"> <li>Taking attributes such as functionality, type of materials, and manufacturing processes into account</li> <li>Allowing variety</li> <li>Improving cost estimation approach</li> </ul>
TCCI, CI	<ul style="list-style-type: none"> <li>TCCI is a normalized version of the DCI</li> <li>Component based</li> <li>Fixed boundaries (0–1)</li> </ul>	<ul style="list-style-type: none"> <li>Focusing only on the percentage of common/unique components rather than cost factors</li> <li>Not efficient in handling large number of options</li> </ul>	<ul style="list-style-type: none"> <li>Weighing components</li> <li>Considering variety in addition to commonality</li> <li>Using more clarification in criteria for differentiation</li> </ul>
PCI, %C	<ul style="list-style-type: none"> <li>Component based</li> <li>Fixed boundaries (0–100)</li> <li>%C is applicable to platform commonality measurement</li> </ul>	<ul style="list-style-type: none"> <li>Unable to capture effect of component costs on commonality</li> <li>PCI assumes known degree of differentiation</li> </ul>	<ul style="list-style-type: none"> <li>Broadening the areas of consideration such as differentiation degree</li> <li>Clarifying differences among differentiating and non-differentiating components</li> </ul>
CDI	<ul style="list-style-type: none"> <li>Fixed boundaries (0–1)</li> <li>Assessing both commonality and diversity</li> <li>Allowing analysis in function, component, and family levels</li> </ul>	<ul style="list-style-type: none"> <li>Ignoring the information about component costs, materials, and processes</li> </ul>	<ul style="list-style-type: none"> <li>Integrating higher architecture levels</li> <li>Using further criteria such as cost and manufacturing issues</li> </ul>
CMC	<ul style="list-style-type: none"> <li>Fixed boundaries (0–1)</li> <li>Considering the effect of each component on the overall commonality level</li> <li>Allowing desired variety/commonality</li> </ul>	<ul style="list-style-type: none"> <li>More information sensitive than other indices</li> </ul>	<ul style="list-style-type: none"> <li>Taking component performance into account</li> </ul>
TCM	<ul style="list-style-type: none"> <li>Using GBOM for incorporating all possible variants into consideration</li> <li>Using past data about selection probabilities of variants</li> </ul>	<ul style="list-style-type: none"> <li>Applicable only if a GBOM is available</li> <li>Difficulty in data collection</li> <li>Better for assemble-to-order systems</li> </ul>	<ul style="list-style-type: none"> <li>Integrating process commonalization studies into the index</li> </ul>

into consideration brings more vulnerability into such indices, due to the increasing dependency on information such as cost estimation. As a result, though other indices might be misleading for large-scale family assessments, new indices also have limitations such as inapplicability to all types of systems (e.g., modular and scalable).

The equations representing the newly developed indices are listed in Table 1.4.

Each index should be carefully selected for use, depending on the family under consideration or characteristics such as module orientation, cost data availability, and assembly considerations.

### ***1.4.2 Family and Platform Configuration and Optimization***

Industry needs and essential concerns in platform design and development are discussed by Simpson et al. (2005), considering both academic developments and industrial cases. Their study addressed key factors for developing robust and flexible platforms and modular architectures with standard interface. Among these factors, formation of cross-functional development teams, along with the ability for continuous improvement based on learned lessons, is noted.

Platform development is shown to be affected by the nature of the product families (i.e., homogenous versus heterogeneous families and functions) and that certain platform leveraging strategies might not be appropriate for all product families (Alizon et al. 2010). Through a proposed metric in their study, called homogeneity versus heterogeneity ratio (HHR), selection of a suitable platform leveraging strategy and identification of the sufficiency of differentiation in a family have been facilitated.

The studies in this area can be categorized into module selection, platform selection, commonality and component sharing, optimization studies, and other exploration studies such as comparisons and application of existing approaches. Table 1.5 shows the studies related to platform selection, variant selection, component sharing, and module selection.

In general, the platform configuration and portfolio optimization problems are mostly limited to cost considerations and modularization based on criteria such as similarity in shape or functions. More considerations can be taken into account at the same time, in order to provide robust designs. A recent study by Emmatty and Sarmah (2012) takes advantage of different product development methodologies (i.e., function-based modular product architecture, platform-based design, and design for manufacture and assembly) to reduce manufacturing costs, incorporate customer requirements, and decrease the throughput time of development. More research works similar to such study seem to be required in the future. Also Simpson et al. (2012) in a recent study have proposed an integrated family design framework, which includes a number of the family design methods developed in the past (i.e., market segmentation grid, DSMs, GVI, commonality indices, and

**Table 1.4** (a) The CMC index, (b) the CDI index, (c) the TCI index

	$(a) CMC = \frac{\sum_{i=1}^p n_i * (C_i^{max} - C_i)}{\sum_{i=1}^p n_i * (C_i^{max} - C_i^{min})} * \prod_{i=1}^p \frac{f_{i1}}{f_{i2}} \quad (1.1)$		
$p$	The total number of components	$C_i^{min}$	The minimum total cost for component $i$
$n_i$	The number of products in the product family that have component $i$	$C_i^{max}$	The maximum total component cost
$f_{i1}$	The ratio of the greatest number of products that share component $i$ with identical size and shape to the number of products that have component $i$ ( $n_i$ )	$f_{i1}^{max}$	The ratio of the greatest number of products that share component $i$ with identical size and shape to the greatest possible products that could have shared component $i$ with identical size and shape schemes
$f_{i2}$	The ratio of the greatest number of products that share component $i$ with identical materials to the number of products that have component $i$ ( $n_i$ )	$f_{i2}^{max}$	The ratio of the greatest number of products that share component $i$ with identical materials to the greatest possible number of products that could have shared component $i$ with identical materials
$f_{i3}$	The ratio of the greatest number of products that share component $i$ with identical manufacturing processes to the number of products that have component $i$ ( $n_i$ )	$f_{i3}^{max}$	The ratio of the greatest number of products that share component $i$ with identical manufacturing processes to the greatest possible number of products that could have shared component $i$ with identical manufacturing processes
$f_{i4}$	The ratio of the greatest number of products that share component $i$ with identical assembly and fastening schemes to the number of products that have component $i$ ( $n_i$ )	$f_{i4}^{max}$	The ratio of the greatest number of products that share component $i$ with identical assembly and fastening schemes to the greatest possible number of products that could have shared component $i$ with identical assembly and fastening schemes
$C_i$	Current total cost for component $i$		

(continued)

Table 1.4 (continued)

(b)	$CDI_{family_p} = \frac{1}{F} \sum_{i=1}^F \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \frac{1}{G_{ik}} \times \sum_{m=1}^{G_{ik}} \left( 1 - \frac{non\_allowed\_com\_div_{i,kgm}}{\max div_{i,kgm}} \right)$ (1.2)	
$F$	Number of functions in the family	$non\_allowed\_com\_div_{i,kgm}$ Non-allowed commonality/diversity for subgroup $g_m$
$k_{ij}$	Component j of function i	$div_{i,kgm}$ Ideal maximum diversity for subgroup $g_m$
$G_{ik}$	Subgroup k of components of function i	$\max div_{i,kgm}$
(c)	$TCI = \left\{ \frac{1}{n} \sum_{j=1}^n \prod_{k=1}^{h_j} (N_k)_{h_j} \sum_{j=1}^n \frac{w_j^2}{n_j} \right\} + \left\{ \frac{\Delta}{m} \sum_{i=1}^n \alpha_i \prod_{k=1}^{h_i} (N_k)_{h_i} \sum_{j=1}^n \frac{w_j^2}{m_j} \right\}$ (1.3)	
$n$	The number of must-generic items	$n_{h_i}$ The number of nodes on path $h_i$ from must- or can-generic item to end product
$m$	The total number of option categories (can-generic items)	$(N_k)_{h_i}$ The quantity per operation of node required by its immediate parent node along path $h_i$
$m_i$	The number of variations in an option category	$A$ The probability that the end product is equipped with an option
$h_i$	The path in the generic bill of materials from node i to end product	$\alpha_i$ The conditional probability that the end product is equipped with option category, knowing that the end product is equipped with an option

**Table 1.5** Studies about platform selection and design

References	Subject	Remarks
Fellini et al. (2004, 2005)	Platform selection	<ul style="list-style-type: none"> <li>• Determination of possible component candidates for sharing</li> <li>• Formulation as relaxed combinatorial problem with maximized component sharing</li> </ul>
Dai and Scott (2007), Liu et al. (2011)	Multiple-platform product family design	<ul style="list-style-type: none"> <li>• Using sensitivity analysis and cluster analysis to improve cost savings due to commonality in Dai and Scott (2007)</li> <li>• Applying a modified GA algorithm to generate alternative product family solutions with different levels of commonality in Liu et al. (2011)</li> </ul>
Ma et al. (2011)	Flexible platform design	<ul style="list-style-type: none"> <li>• Developing design process based on flexible product platform with parametric design</li> </ul>
Michalek et al. (2006)	Product line profitability maximization	<ul style="list-style-type: none"> <li>• Integration of marketing and design perspectives</li> <li>• Assessing production functionality, manufacturing costs, and performance of market at the same time to maximize profit of product line solution</li> </ul>
Jiao et al. (2007e)	Complex and interrelated product family design optimization	<ul style="list-style-type: none"> <li>• Using GBOM and generic genetic algorithm for finding the optimal design alternative</li> <li>• Considering the maximum customer-perceived cost/benefit ratio</li> <li>• Equipped with a hybrid-constant handling strategy and complexity analyses</li> <li>• Applicable to PFD problems with complex data and inherent interrelationships</li> </ul>
Jariri and Zegordi (2008)	Platform design optimization	<ul style="list-style-type: none"> <li>• Developing a mathematical programming approach using QFD</li> </ul>
Kim et al. (2006)	Integrated function module optimization and platform selection	<ul style="list-style-type: none"> <li>• Using QFD-based multi-attribute optimization model</li> </ul>
Turner et al. (2011)	Comparison of a top-down platform to a bottom-up platform	<ul style="list-style-type: none"> <li>• Top-down platform: leveraging heterogeneous discrete choice models</li> <li>• Bottom-up platform: designing a varying number of products independently</li> <li>• Simultaneous optimization of the commonality and preference share</li> </ul>
Qu et al. (2011)	Guidelines for product platform development	<p>Assisting platform-based product development through discussing guidelines about:</p> <ol style="list-style-type: none"> <li>1. Ways to establish platforms for a family of products in a given industrial and market context</li> </ol>

(continued)



**Table 1.5** (continued)

References	Subject	Remarks
		2. How to choose and customize the most appropriate product platform to meet the customer requirements
Pasche et al. (2011)	Effects of platforms on new product development projects	Investigating the effects of platforms on new product development projects
Sköld and Karlsson (2012)	Consideration of product platform replacement challenges to managers	Exploring operational challenges for manager, when platforms need to be replaced

optimization) in order to facilitate identification of commonality specifications based on translation of customer needs.

Table 1.6 shows the studies regarding variant selection, modularization techniques, and other studies regarding modularity.

However, more issues than module identification and component modularization exist in the platform design areas. This field involves additional considerations such as what components should be selected for platform formation, what are the optimal values for platform/non-platform parameters, and what is the best level of modularity/commonality among variants.

Platform design problems are usually split into two stages, the first dealing with finding the values of platform variables and the second optimizing variables for each variant in the family under desired targets subject to existing constraints (Dai and Scott 2006). Several approaches have been developed over the past decade for product family and platform optimization (Khajavirad et al. 2009), and forty of them have been reviewed and classified in Simpson (2005). According to his review and the work of Fujita (2002), platform design optimization problems are mainly categorized into three classes: (i) a class of problems in which the platform configuration is assumed as known and the objective is to find the optimum design variables and product attributes for that particular configuration (i.e., parametric optimization); (ii) a class of problems in which the objective is to select optimal modules among a number of predetermined modules, where the module's design is already known (i.e., structural optimization); and (iii) a class of problems that aim to find the platform configuration and the optimal product attributes simultaneously. Table 1.7 shows the studies related to optimization of families and platforms, belonging to the first or second class. In this third class, decision can be made on swapping any set of modules, scaling some modules to satisfy specific requirements, or both (Li et al. 2007). Several studies have proposed approaches for handling the problems particularly in the first two classes.

For the last class, the decision about each issue is dependent on the other, the computational complexity is increased for such problems, and simultaneously optimization issues will be inevitable (Khajavirad et al. 2009).

In a recent study, commonality of the baseline product which is the product from which the other variants can be differentiated is balanced with performance of

**Table 1.6** Studies about variant selection, commonality, and modularity

References	Subject	Remarks
Zacharias and Yassine (2008)	Variant selection	<ul style="list-style-type: none"> <li>Using commonality as a performance measure to determine the best amount of investment on platform</li> </ul>
Meng et al. (2007)	Module identification	<ul style="list-style-type: none"> <li>Multi-objective GA for identifying the isolated components that need to enter a module</li> </ul>
Torstenfelt and Klarbring (2006, 2007)	Module optimization	<ul style="list-style-type: none"> <li>Simultaneous optimization of size, shape, and topology for reducing time to market</li> </ul>
Yang and Wang (2009)	Modular design process improvement	<ul style="list-style-type: none"> <li>Considering correlations among functional, structural, and behavioral aspects of the product family, through <i>relationship constraint network</i> (RCN)</li> </ul>
Johnson and Kirchain (2010)	Linking commonality metrics to cost savings	<ul style="list-style-type: none"> <li>Leveraging process-based cost models for projecting the cost reductions of sharing</li> </ul>
Yu et al. (2007)	Partitioning architecture into modules	<ul style="list-style-type: none"> <li>Using <i>GA</i> for clustering modules</li> </ul>
Li et al. (2012)	Modular architecture design	<ul style="list-style-type: none"> <li>Proposing an integrated product modularization scheme based on flow analysis, design structure matrix (DSM), and fuzzy clustering</li> </ul>
Fellini et al. (2006)	Commonality selection	<ul style="list-style-type: none"> <li>Combining the two approaches developed in Fellini et al. (2004, 2005)</li> <li>Problem size reduction in platform selection</li> <li>Sensitivity analysis for obtaining performance loss bounds</li> <li>Maximizing commonality among variants and minimizing individual performance deviations</li> </ul>
Arciniegas and Kim (2011)	Guidance to component sharing	<ul style="list-style-type: none"> <li>Finding the best matches among components, using a <i>DSM</i> and architecture information</li> <li>Module definitions through two concepts: <ul style="list-style-type: none"> <li><i>Impact metric</i> (IM) for describing the easiness of changing a component in a given architecture</li> <li><i>Minimum description length</i> (MDL) for optimal clustering</li> </ul> </li> </ul>
Arciniegas and Kim (2012)	Component sharing and architecture optimization	<ul style="list-style-type: none"> <li>Finding optimal clustering and component sharing strategy for case of sensitive elements with security requirements</li> </ul>
Agrawal et al. (2012)	Hybrid model of component sharing and platform modularity	<ul style="list-style-type: none"> <li>Seeking the most profitable design which takes advantage of both modularity and integrality</li> </ul>
Jiao et al. (2007a)	Process platform development	<ul style="list-style-type: none"> <li>Discussing managerial implications of process platforms</li> </ul>

(continued)

**Table 1.6** (continued)

References	Subject	Remarks
Seol et al. (2007)	Design process modularization	<ul style="list-style-type: none"> <li>• Providing guidelines to structure the design process and alleviating the managerial complexity associated with design</li> </ul>
Khire et al. (2008)	Comparison of commonality selection methods	<ul style="list-style-type: none"> <li>• Discussing the challenges of family optimization problem formulation and design space exploration</li> <li>• Visualization is recommended for bridging the chosen commonality to its according performance loss</li> </ul>

potential differentiated variants at the same time, through a GA-based optimization approach (Chen and Martinez 2012). This is a new look to this area, as it does not require prior determination of specifications for the baseline product, and as a result, it can help in a more efficient design for variants.

Recent research studies are moving toward handling the third class of problems, often referred to as “joint product platform selection and design,” and they attempt to provide effective optimization algorithms for problems with many variants. The works related to issues in optimization of this class are summarized in Table 1.8.

The level of commonality has been restricted in most of these studies so that the components should either be shared among all variants or be unique for each variant [all-or-none, e.g., see Khire et al. (2006)]. While aimed at reducing computational complexity, this assumption imposes limitations on the optimization problem and sometimes results in impractical or suboptimal solutions (Thevenot and Simpson 2006).

In addition to the necessity of relaxing the all-or-none commonality assumption, there are more requirements for properly handling and solving optimization problems of this class. For example, mixed variables impose more complexity on the solution procedure. One solution for avoiding such complexity is to use approximation in the continuous space.

There are still plenty of opportunities for improvement in simultaneous optimization of the platform and non-platform parameters and platform selection. For example, including more factors into consideration offers both benefits and limitations. It is beneficial because the obtained solutions will be more reliable due to simultaneous consideration of all possible important factors in design, but it adds to the complexity of the problem, and it might result in difficulties for analysis and optimization.

Also, there are assumptions that need to be replaced by realistic conditions; however, such considerations come at the expense of added complexity. Therefore, development of algorithms and approaches for handling such complexity can lead to remarkable advancements in platform configuration and optimization, particularly for the third class of optimization problems. Tools and concepts such as GBOM, cost models, customer satisfaction assessments, and other concepts can also be helpful in broadening consideration and optimizing multiple objectives. Such tools can help in obtaining robust design solutions if they are considered along with other technical objectives such as commonality level and manufacturing flexibility.

**Table 1.7** Studies about optimization of families and platforms for Classes (i) and (ii)

Reference	Subject	Remarks
Michalek et al. (2006)	Product line optimization	<ul style="list-style-type: none"> <li>• Considering market demand and manufacturing</li> <li>• Decomposing the product line through ATC</li> <li>• Considering manufacturing equipment sharing (rather than component sharing)</li> </ul>
Tucker and Kim (2008)	Product portfolio optimization	<ul style="list-style-type: none"> <li>• Using <i>data mining techniques</i> and introducing a customer's maximum purchasing price, for controlling the level of engineering design according to customers' preferences</li> <li>• Applying ATC and <i>analytical target setting (ATS)</i> to maximize the profit in a product family architecture reconfiguration problem</li> </ul>
Belloni et al. (2008)	Exploration of advancements in product line optimization for large problems	<ul style="list-style-type: none"> <li>• Comparison between a vast number of current heuristic optimization approaches</li> </ul> <p>Results confirm that genetic algorithm, simulated annealing, divide-and-conquer, and product-swapping approach perform particularly well and can obtain near-optimal solutions even if measurement errors exist</p>
Wäppling et al. (2011)	Integrated product requirement optimization and design optimization	<ul style="list-style-type: none"> <li>• Maximizing product performance and minimizing total costs</li> <li>• Using multi-objective optimization to obtain the best trade-off among requirement specifications, design choices, and cost</li> <li>• Integrating the product requirement specification with the design objectives to reduce problem complexity</li> </ul>
Thevenot et al. (2007)	Product line selection through multi-attribute utility function	<ul style="list-style-type: none"> <li>• Discussing engineering applications of the developed model</li> </ul>
Perez and Linsey (2011)	Exploring the principles of product scaling	<p>Six scaling principles have been identified:</p> <ul style="list-style-type: none"> <li>• Changing the energy source</li> <li>• Simplifying the system</li> <li>• Changing the method</li> <li>• Combining functions</li> <li>• Changing the parameters</li> <li>• Transferring components/features directly from the original scale to the desired one</li> </ul>
Ong et al. (2006)	Design requirement optimization	Applying <i>multi-objective optimization struggling GA</i> to a cost model

### 1.4.3 Metrics for Design and Assessment of Platforms

According to Scott et al. (2006), there is no consensus on the best approach or criteria to solve problems that need commonality selection among variants of family. Many criteria have been used to help in determination of components to

**Table 1.8** (a) Studies about joint product platform selection and design, (b) other studies about platform selection and configuration

Reference	Subject	Remarks
(a)		
Freeman et al. (2011)	Platforms' identification among component subspaces	<ul style="list-style-type: none"> <li>Fuzzy clustering techniques for component grouping</li> </ul>
Khajavirad and Michalek (2008)	Platform optimization in continuous design space	<ul style="list-style-type: none"> <li>Extending the CI by Martin and Ishii (1996) and Martin and Ishii (1997), to handle continuous design spaces</li> <li>Leveraging gradient-based optimization methods</li> </ul>
Khajavirad and Michalek (2007)	Joint platform selection and design	<ul style="list-style-type: none"> <li>Using (ATC) for decomposing the joint problem</li> <li>Determining the optimal platform configuration at the system level and the individual variants at the subsystem level</li> </ul>
Khajavirad and Michalek (2008)	Joint problem optimization	<ul style="list-style-type: none"> <li>Using decomposed gradient-based optimization</li> <li>Assumptions <ul style="list-style-type: none"> <li>Generalized commonality</li> <li>Relaxation into continuous design space</li> <li>Using single-stage decomposed optimization</li> </ul> </li> </ul>
Khajavirad et al. (2007)	Joint problem optimization	<ul style="list-style-type: none"> <li>Using decomposed multi-objective genetic algorithm with generalized commonality</li> <li>Introducing a two-dimensional commonality chromosome with two levels: <ul style="list-style-type: none"> <li>The upper level finds the optimal platform configuration</li> <li>The lower level optimizes variants of the family</li> </ul> </li> </ul>
Khire et al. (2006)	Joint platform design	<ul style="list-style-type: none"> <li>Applying <i>selection-integrated optimization method</i> (SIO)</li> <li>Using gradient-based method</li> <li>Continuous relaxations for solving this combinatorial problem</li> </ul>
Luo et al. (2008, 2009)	Simultaneous optimization of platform and non-platform engineering characteristics	<ul style="list-style-type: none"> <li>Maximizing overall customer satisfaction through the variants and minimizing quality loss caused by commonality</li> <li>Proposing a single-stage linear programming (LP) optimization approach to solve a similar problem</li> </ul>

(continued)

**Table 1.8** (continued)

Reference	Subject	Remarks
Dai and Scott (2006)	Effective product family and platform design	<ul style="list-style-type: none"> <li>• Applying three approaches to platform design:               <ul style="list-style-type: none"> <li>– <i>Preference aggregation</i> for aggregating multiple objectives into a single objective function</li> <li>– <i>Min-max method</i> for fulfilling the performance ranges</li> <li>– <i>Single-stage optimization</i> of platform and non-platform parameters</li> </ul> </li> </ul>
Gao et al. (2009)	Simultaneous optimization of platform and non-platform parameters	<ul style="list-style-type: none"> <li>• Finding the optimal platform and non-platform parameters in a modularized scale-based platform</li> </ul>
Moon et al. (2008)	Platform optimization through <i>dynamic multi-agent system</i> (DMAS)	<ul style="list-style-type: none"> <li>• Finding the optimal platform through decomposing PFD tasks and assigning them to agents via a mathematical model</li> </ul>
(b)		
Li and Huang (2009)	Adaptive product family design	<ul style="list-style-type: none"> <li>• Applying a multi-objective evolutionary algorithm based on non-dominated sorting genetic algorithm (NSGA-II), used in Khajavirad et al. (2009) and Khire et al. (2008)</li> <li>• Allowing commonality at different levels (e.g., product, module, component, and parameters)</li> <li>• Using GBOM and IBOM for representing the products</li> </ul>
Li et al. (2007)	Adaptive design of platform	<ul style="list-style-type: none"> <li>• Introducing an approach called interwoven evolutionary algorithm (IEA)</li> <li>• Combining genetic operators from genetic programming (GP) and genetic algorithms (GA) into one evolutionary process</li> <li>• Optimizing individual structures first and then parametric executing optimization through the GA</li> <li>• Comparing the performance of this approach to a previously developed algorithm (Tandem Evolutionary Algorithm by Huang et al. (2007) in which the GA for parametric optimization was nested into GP for structural optimization)</li> <li>• It was shown that IEA outperforms TEA in terms of computational time and efficiency</li> </ul>

(continued)

**Table 1.8** (continued)

Reference	Subject	Remarks
Chen and Wang (2008)	Multiple-platform selection	<ul style="list-style-type: none"> <li>Using fuzzy clustering and a two-level chromosome GA</li> <li>Assuming predetermined commonality level of the platform</li> </ul>
Khajavirad and Michalek (2009)	Deterministic global optimization of non-convex quasi-separable MINLP problems	<ul style="list-style-type: none"> <li>Applicable to platform design problems</li> <li>The deterministic approach is shown to be of higher quality than the stochastic local search approaches and of faster convergence</li> </ul>
Moon et al. (2011)	Multi-objective platform optimization	<ul style="list-style-type: none"> <li>Proposing a multi-objective particle swarm optimization (MOPSO)</li> </ul>
Khire and Messac (2008)	Adaptive design of platform	<ul style="list-style-type: none"> <li>Using SIO to simultaneously identify the adaptive design variables (those that can be changed during later stages of life cycle) and fixed variables</li> </ul>
Li et al. (2008)	Structural and parametric optimization and customization of platform	<ul style="list-style-type: none"> <li>Proposing an evolutionary algorithm in platform product customization problem</li> </ul>
Öman and Nilsson (2011)	Critical constraint approach for structural design	<ul style="list-style-type: none"> <li>Applying critical constraint method (CCM) for optimization of structural product families</li> </ul>

share within a family (Arciniegas and Kim 2011). Cost considerations have been the main concern in regard with family design and platform development, but a number of new criteria have emerged recently to support design decisions: bill of materials (Steva et al. 2006), platform investment (Moon et al. 2007; Zacharias and Yassine 2007), environmental concerns (Pandey and Thurston 2008), and product design variables (Khajavirad and Michalek 2008; Khajavirad et al. 2009).

Park and Simpson (2008) developed a cost estimation framework to consider the effect of design decisions on activity costs. They mapped activity costs to product family components based on an activity-based costing system (ABC) for obtaining cost-effective product family design. Johnson and Kirchain (2009) applied a process-based costing model to assess the economic effects of component (or material) sharing. They showed that due to interdependencies between components (materials), their coupling can result in remarkable savings in design.

Otto and Hölttä-Otto (2007) introduced a framework that used six metrics for multi-criteria platform evaluation involving customer, variety, flexibility, complexity, organization, and after-sale. Their proposed framework helps to guide the early stages of platform architecture identification and was shown to be useful for refinement of existing platforms. Also, Hölttä-Otto and de Weck (2007) assessed the effect of technological constraints on integration of design and architecture. They defined two metrics: (1) the nonzero fraction (NZF) that captures the degree of sparse interrelationships between components and (2) the singular value modularity index (SMI), which captures the degree of internal coupling.

SMI allows for analyzing the degree of modularity for any architecture independent of subjective choices of modules. Both metrics take values between zero and one, and they are applied to two product pairs that are functionally equivalent but different in terms of technical constraints in their study. The results of their study support the hypothesis of modularity dependency to technological constraints and that some products inherently have less modularity due to technological factors.

Reviewing five main research studies about platform design under uncertainty, Suh et al. (2007) showed that their approach for designing flexible platforms outperformed the previous works, due to mapping uncertainty into product attributes, design variables, physical components, and even into relevant evaluation costs. However, assuming a single platform for all variants is one of their study limitations for the cases with remarkable differences among variants.

A study by Blecker et al. (2006) proposed a key metrics-based approach for controlling the adverse effect of variety-induced complexity in mass customization. They integrated the proposed metrics into a comprehensive system, including the component correlations for providing a better tool for controlling complexities. The other applied criteria can be listed briefly as follows:

- Using extended QFD (Zhang et al. 2006a; Li et al. 2006) to facilitate platform design and to develop a process model for variants with the same functions but with different parameters.
- Developing cost model for variety analysis (Helo et al. 2009) to balance the cost of adding variants to the BOM and the cost reduction resulting from commonality.
- Developing a platform construction method for large-scale families (Wang et al. 2007a) with the objective of minimizing development cost and performance loss costs.
- Developing *product family evaluation graph approach* (PFEG) (Ye and Gershenson 2008) for assessing different design alternatives based on the trade-off between commonality and variety among product attributes.
- Simultaneous consideration of commonality and variety through using redefined indices (CMC and CDI) (Ye et al. 2009) to identify the best design alternatives and to link marketing and engineering design decisions during PFD.
- Leveraging data mining techniques for platform and product family development. Examples: developing a *process family* by identifying similarities among existing facilities and manufacturing processes (Jiao et al. 2007a, d); developing a PFD methodology for identifying product variant attributes (Tucker et al. 2010); mapping customer needs (CNs) to functional requirements (FRs) by considering correlations of CNs and FRs through integration with QFD and based on historical data (Qin and Wei 2009); leveraging decision trees for searching design options through narrowing down the large sets of customer preferences data into a limited number of design concepts by Tucker and Kim (2007); and identifying and categorizing modules based on functional hierarchies and their grouping based on level of similarity by Moon et al. (2010), using rule mining and knowledge discovery techniques along with data mining.



- Using fuzzy clustering techniques to identify modules in product architectures (Moon et al. 2006a); fuzzy clustering algorithm for developing functional expressions for products and associating rule mining techniques for investigating relation of CNs to product components in portfolio identification (Zhong and Zhong 2006); considering customer needs and fuzzy clustering at the same time (Xu et al. 2006) for identifying proper module partitioning schemes that balance structural and functional relativities among the components of a family.
- Using *axiomatic design* (Jiang et al. 2007) to decompose design tasks into hierarchies of functions and structures, with the objective of minimizing interdependencies among architecture modules.
- Platform exploration and identification (Steva et al. 2006) through combining *BOM*, *DSM*, *function diagrams*, and *functional-component matrices* for in-depth analysis of product sets.

In addition to assessing performance of newly developed approaches, it is useful to study and examine previous practical instances of successful platforms. Ford's Model T is an example of such a platform, which was studied by Alizon et al. (2009b) to identify success factors in customizing and developing many variants from it. Among the factors identified by their study, the following items can be addressed: late differentiation of the car body, development of common interfaces among platform and variant car bodies, platform evolution through reuse, modular design for achieving different models, scalable design, customized outsourcing (i.e., allowing the customers to take care of assembling the customized gadgets and involving the customer in the design process), vertical integration rather than outsourcing to prevent complexity in platform-based production, mass customization through using independent platforms and car bodies, complete redesign of products in their final life cycle stages instead of tiny changes, and, finally, parallel customization (i.e., using specialized manufacturers, for tailoring final products for specific markets, but focusing on the core of products inside their own manufacturing territory).

In summary, while a diverse set of metrics and criteria have been developed and applied to commonality and component sharing decisions, it seems less attention has been paid to the following:

- *Simultaneous consideration of commonality and diversity*: Though this issue is shown to be helpful in efficient design of product variants in a family and resolving the trade-off between commonality and diversity, it has rarely been addressed in the studies relevant to commonality selection.
- *Variation and uncertainty involvement*: Very few studies have involved uncertainty consideration in PDF decisions as concluded from this review. However, research shows that many factors that affect optimality of design might change over time and therefore uncertainty can endanger the selected design solutions which should be taken into consideration when obtaining a robust platform.
- *Correlations and dependencies*: Many of the factors affecting optimal design might be affected by one another, and such correlations may result in suboptimal design solutions if not considered properly. They also may result in design

solutions with higher costs and performance sacrifices. This vulnerability can be controlled through a careful consideration of possible dependencies and correlations among the factors that affect optimality of design options.

### ***1.4.4 Design Support Systems***

One of the powerful tools for facilitating product family design, platform configuration, and family redesign is design and decision support tools that allow for integrating, sharing, browsing, reusing, and classifying design information and knowledge. In addition, standardization and unification of design knowledge and information pieces are possible with the aid of such tools, which results in creating valuable resources for design improvements and for future developments.

The developed design and decision support systems mostly have exploited the product family architecture, design structure matrix, or other relation matrices as a basis for evaluating the dependencies, design information, and for providing solutions and suggestions for better design options.

A comprehensive terminology for PFD processes is developed by Alizon et al. (2007) by considering platform and family development drivers and by using two existing platform development approaches: top-down and bottom-up development. In their study, based on concurrent engineering principles, four processes are proposed for facilitating product family development, with the objective of embodying consistency among family development processes, and controlling additional time and cost challenges which accompany PFD. The summary of the studies providing techniques and approaches for assisting design is provided in Table 1.9.

## **1.5 Back-End Issues of Product Family Design**

### ***1.5.1 Reconfigurability of the Design***

Reconfigurable system design attempts to change the configuration of a family within an acceptable time frame and cost to meet multiple functional requirements and to operate under varying conditions. This approach is shown to help achieve a product family with desired diversity and commonality that can meet a limited budget as well (Siddiqi et al. 2006).

Reconfigurable system design, flexible manufacturing, and product family design principles were investigated by Cormier et al. (2009) in order to identify key success factors in design for reconfigurability. More studies about reconfigurable system design can be found in Siddiqi et al. (2006), Ferguson et al. (2007a, b), Ferguson and Lewis (2008), Anzanello and Fogliatto (2011), and Abdi (2012). Since reconfigurability requires considerations in large scale, complexity comes as an expense to it; therefore, *multidisciplinary design*

**Table 1.9** Studies about design support systems and techniques

References	Subject	Remarks
Cutting-Decelle et al. (2007)	Developing ISO 15531 MANDATE standard for managing modularity	Exploiting international standards for exchanging industrial manufacturing management data
Williams et al. (2007)	Utility-based compromise decision support	<ul style="list-style-type: none"> <li>Proposing a Product Platform Constructal Theory Method (PPCTM) for enabling configuration of platforms for customized product development</li> <li>Handling multiple commonality level, multiple product specifications, and inherent trade-offs between platform extent and performance</li> <li>Handling multiple objectives and non-uniform demand of variants</li> </ul>
Liang and Huang (2002)	Agent-based design support	<ul style="list-style-type: none"> <li>Enabling integration of information in complex networks for platform module selection</li> </ul>
Moon et al. (2006b)	Multi-agent design decision support	<ul style="list-style-type: none"> <li>Considering market mechanisms with the objective of dismissing volatile modules, or selecting stable ones</li> </ul>
Nomaguchi et al. (2006), Zha and Sriram (2006)	Knowledge management-oriented design support system	<ul style="list-style-type: none"> <li>Facilitating product family architecture design and modeling</li> <li>Providing a design repository without redundant data</li> </ul>
Shamsuzzoha and Helo (2012)	Modular architecture design assistance	<ul style="list-style-type: none"> <li>Investigate the importance of information management for modular product architecture</li> </ul>
Baxter et al. (2007)	Knowledge reuse methodology	<ul style="list-style-type: none"> <li>Providing an integrated framework for storing, retrieving, and applying the design knowledge in early stages of design</li> </ul>
Liu and Özer (2009)	Decision framework for maximizing product replacement profits	<ul style="list-style-type: none"> <li>Suggesting that product replacement would be beneficial only if its performance gap is above a defined threshold</li> <li>Concluding that in fast-changing industries attempting to reduce replacement costs is an essential factor for compensating possible inefficiencies of frequent replacements and such cost reduction can be obtained through platform development and modular design</li> </ul>
Wang et al. (2007a)	Decision support model to assist product family configuration	<ul style="list-style-type: none"> <li>Assessing the relations between changing parts (product variety) and life cycle costs (LCC)</li> </ul>
Allada et al. (2006), Ye et al. (2009)	Product platform problem taxonomy	<ul style="list-style-type: none"> <li>Classification and identification of benchmark problems</li> </ul>
Hoogeweegen et al. (2006)	Trends assessment in multi-player business networking	<ul style="list-style-type: none"> <li>Using a simulation game for studying the changing nature of business networks when production goes toward mass customization</li> </ul>

(continued)

**Table 1.9** (continued)

References	Subject	Remarks
Xu et al. (2008)	Family architecture evolution support	<ul style="list-style-type: none"> <li>Proposing TRIZ evolution theory of technologies to facilitate PFA evolution</li> </ul>
Lim et al. (2011)	Ontology-based representation for design schemas	<ul style="list-style-type: none"> <li>Modeling complex interrelationships among components and other non-component-based information</li> <li>Proposing a faceted framework for searching and retrieving information of product family design</li> </ul>

*optimization* (MDO) has been of interest as a solution for obtaining the desired performance and determining optimal configuration in such systems. Ferguson et al. (2009) developed one such MDO method for finding the core architecture for a product family, and their proposed architecture accommodates a number of changing design variables. In their study, an MDO approach is leveraged to optimize a number of subsystem level variables along with some system-level ones.

Since product family configuration is affected by many external and internal factors (i.e., customer demand, market segmentation, manufacturing capabilities, company size, and management policies), and as it is a complex issue due to variation of objectives over time, reconfigurability can help in obtaining design options with desired performances. However, reconfigurability is usually accompanied by additional costs. Therefore, developing more effective approaches for handling contradictory objectives with consideration of constraints can result in advancements in PFD.

### ***1.5.2 Redesign and Design Reuse Strategies***

The bottom-up approach in product family design deals with redesign of existing variants or design reuse strategies for efficiency improvement and cost reduction. While reuse of designs for components, modules, and functions is a common approach used in product family and platform development, it might turn into a risky policy when change occurs in technology, customer expectations, and consequently in product life cycle.

According to a study by Shooter et al. (2007), some key factors in the success of product family and platform development are time and cost saving by reusing design of available components, using similar design approaches and manufacturing processes for separate products that are produced on different platforms, benefiting from platform development for further offerings, and deliberately leaving opportunities for future customization. Table 1.10 summarizes the research about reuse and redesign for platforms and product families.

**Table 1.10** Studies about design reuse and redesign for platforms and families

References	Subject	Remarks
Thevenot and Simpson (2006)	Development of a product family redesign framework	<ul style="list-style-type: none"> <li>• Developing the framework based on comprehensive evaluation of six commonality indices</li> <li>• The proposed framework provides valuable insights into the relationships between different leveraging strategies and resulting levels of commonality</li> </ul>
Suh et al. (2007)	Platform redesign process development	<ul style="list-style-type: none"> <li>• Considering uncertainty and emphasizing differentiating parts (i.e., the unique parts which result in variation) as potential options for redesign or modification</li> <li>• Considering such parts as opportunities for making flexible platforms</li> </ul>
Yan et al. (2007), Yan and Stewart (2010)	Generation of design reuse solutions	<ul style="list-style-type: none"> <li>• Developing a methodology named GeMoCURE to modularize a product family</li> <li>• Generating solutions with the greatest possible variation and minimum costs, time, and resources</li> <li>• Applying the GeMoCURE to a number of small and medium-sized enterprises (SMEs)</li> </ul>
Raffaelli et al. (2011)	Software development for redesign decision-making support	<ul style="list-style-type: none"> <li>• Developing a software called Modular for facilitating early assessment of feasibility, costs, and time for any design alternative</li> <li>• The proposed software is based on multilevel representation of products in terms of interrelated functions, modules, assemblies, and components</li> </ul>
Hou et al. (2011)	Research on product family modeling for design resource reuse	<ul style="list-style-type: none"> <li>• Developing a reusable design resources model based on generic BOM</li> </ul>
Ong et al. (2006)	<i>Development of product family design reuse method (PFDR)</i>	<ul style="list-style-type: none"> <li>• Building an architecture based on existing products</li> </ul>
Xu et al. (2007)	Product performance evaluation from reuse perspective	<ul style="list-style-type: none"> <li>• Application of the PFDR approach</li> </ul>
Tao and Yu (2012)	Incorporating reuse and remanufacturing in product family planning	<ul style="list-style-type: none"> <li>• Taking advantage of the differences in technological lifetime of components and consumers' preference regarding the product functionality and quality and environmental performance, for enhancing reuse</li> </ul>

(continued)

**Table 1.10** (continued)

References	Subject	Remarks
Salhieh (2007)	Redesign for homogenizing product families	<ul style="list-style-type: none"> <li>• Reducing redundancies among heterogeneous product portfolios (with minimum or no shared components)</li> <li>• Forming homogenous families which possess physical commonality among the components with similar functions or similar manufacturing processes</li> </ul>
Morrise et al. (2011)	Design of collaborative product	<ul style="list-style-type: none"> <li>• Collaborative products are created by recombining physical components of two or more products temporarily for performing additional tasks</li> <li>• Benefiting from modularity and added functionality</li> </ul>
Kalyanasundaram and Lewis (2011)	Reconfiguration for multifunctionality	<ul style="list-style-type: none"> <li>• Quantifying similarities between functions for common, similar, and unique components among different products</li> <li>• Mapping the function information to the components of the existing products to derive the architecture of the new reconfigurable product</li> </ul>

In order to obtain efficient redesign, one needs reliable prior performance assessment. However, information sufficiency is a remarkable concern of redesign/reuse strategies for both performance assessment and parameter selection. Therefore, further standardization of the design information for different families can assure higher efficiency for these approaches.

### ***1.5.3 Supply Chain Issues for Product Family Design***

The supply chain management is of importance for design and development of product families, in the sense that proper collaboration with external partners allows for more variety, efficiency, and customer-oriented product offerings (Jiao et al. 2007c). The problem of supplier selection and supply chain configuration has been studied from various aspects, including different criteria such as cost, flexible planning of resources, and assembly time. The supply chain planning has been considered from different perspectives as summarized in Table 1.11.

While aspects such as homogenized module allocation, overall supplier selection cost minimization, flexible supply chain design, and time to market minimization have been tackled in regard with supply chain management for product families, there are still opportunities to expand research on supply chain issues

**Table 1.11** Studies in regard to different aspects of supply chain for product families

References	Subject	Remarks
Qian (2009), Agard and Penz (2009), Jin and Chen (2008), Wang et al. (2007b)	Supply chain profit maximization, cost minimization, and cost considerations	<ul style="list-style-type: none"> <li>Stochastic programming for demand uncertainty (Jin and Chen 2008)</li> <li>Including GBOM (Wang et al. 2007c)</li> </ul>
Wang and Ning (2007)	Supply chain model development	<ul style="list-style-type: none"> <li>Standardization of components, processes, procurements, and products over the supply chain</li> </ul>
Das (2011)	Strategic supply chain planning	<ul style="list-style-type: none"> <li>Consideration of various market scenarios which can result in supply uncertainty</li> </ul>
Lamothe et al. (2006), Khalaf et al. (2008)	Supply chain structure optimization	<ul style="list-style-type: none"> <li>BOM optimization in Khalaf et al. (2008)</li> </ul>
Agard et al. (2006), Khalaf et al. (2011)	Average assembly time minimization	<ul style="list-style-type: none"> <li>Simultaneous design of the assembly process and the supply chain (Khalaf et al. 2011)</li> </ul>
Wang (2010)	Outlining the desired characteristics of a supply chain	<ul style="list-style-type: none"> <li>Leveraging complex networks</li> </ul>
Schönsleben(2012)	Studying design-to-order processes	
Fujita et al. (2011, 2012)	Integrating module commonalization decisions with supply chain configuration	<ul style="list-style-type: none"> <li>Simultaneous module commonalization and supply chain configuration for a given product architecture</li> </ul>
Luo et al. (2011), Khalaf et al. (2009)	Integrating <i>supplier selection</i> and product family design	<ul style="list-style-type: none"> <li>Minimizing production and transportation costs under time constraints (Khalaf et al. 2009)</li> </ul>
Shahzad and Hadj-Hamou (2012)	Integrating supply chain and family architecture design for highly customized demand	
Hilletofth et al. (2010)	Coordinating new product development with supply chain management	
Marion et al. (2007)	Evaluating platform commonality sourcing decisions	
Wang and Wang (2009)	Module allocation	<ul style="list-style-type: none"> <li>Product family-oriented supply chains</li> </ul>
Sandborn et al. (2008)	Studying long-term supply chain disruption	<ul style="list-style-type: none"> <li>Considering disruption effect of the reuse ability of parts in platform design</li> </ul>
Mula et al. (2010)	Supply chain planning for fuzzy demand	
Trentin et al. (2011)	Proposing <i>form postponement</i> approaches	
Georgiadis and Athanasiou (2010), Vlachos et al. (2007), You et al. (2010)	Capacity planning studies	

for product family design and platform development. For example, different stages of supply chain management can be involved in design decision-making: uniform resource sharing and allocation, increasing the possibility of late differentiation, exploiting outsourcing benefits, resource/material leveling based on commonalities, techniques to reduce the lead times and the inventory level for higher efficiency and cost saving, etc. Also, integration of the supply chain design and the family design will enable more efficiency that has remarkable potential for further investigations. One study in this regard is done by Khalaf et al. (2011), which applies Tabu search for simultaneous design of the assembly process and supply chain.

Capacity planning is another important issue to be tackled in the back-end stages of product family design, which requires consideration of factors such as life cycle assessments and technological changes for developing the next generation of variants derived from the same family. This area seems to become of interest for further instigation, according to the literature review of recent year studies. Georgiadis and Athanasiou (2010) attempted to investigate capacity planning for recovery and remanufacturing in product families, when a variant is the successor of a previous variant. Other studies about capacity planning that have recently emerged include effective capacity planning (Vlachos et al. 2007) and multisite capacity, production, and distribution planning through multi-period mixed-integer linear programming (You et al. 2010).

## 1.6 Future Research Directions

From this review, the following aspects can be noted as open areas for future research:

*Integration of Management and Engineering Aspects.* Design of product families and platforms requires consideration of their effect on market demand, and without a thorough analysis and examination of market conditions, the resulting design options may not realize the expected benefits.

A number of remarkable studies, addressing this issue particularly for single product design, include Li and Azarm (2000, 2002) and Williams et al. (2010), which have studied product design and development with consideration of uncertainty and market competition, and Michalek et al. (2005, 2007) and Shiau and Michalek (2009a), which have integrated market into engineering design models. Performance assessment of design options in target market segments should be included in product family design decision-making (Kumar et al. 2006). Proper tools and techniques are also required for facilitating performance assessment, which in turn dictates more research for improving this field.

*Platform Divergence.* A related theme emerging in the literature is the challenge of managing commonality through product family design and development. Referred to as *platform divergence* (Boas 2008), recent studies have identified the difficulties



that many companies encounter (1) establishing targets for commonality in a family and (2) enforcing those targets during the product family development process. A recent vehicle development study at Ford exemplifies this case (Montano 2011), and the Joint Strike Fighter provides many lessons for the challenges of this practice (Boas 2008). Considerably more research is needed to understand the roles that designers, engineers, and managers play to effectively address these issues.

*Consideration of Dynamic Issues.* Since market conditions and customer preferences change rapidly, this dynamic issue along with its uncertainties should be included in the design considerations (Zacharias and Yassine 2008). For example, Shiau and Michalek (2009a) have considered the effect of competitors' reactions on pricing new products, and such consideration can also be needed in the product family design arena. Besides, there are parameters such as fluctuations in demand over time, or dependency of demand on price and competitors' offers, which in turn affect optimal pricing policies for newly developed product variants. Such considerations are also worth to be extended to product family design studies, while most are just applied to single product positioning and optimization (Michalek et al. 2006). In addition, technology changes, management policies, supply chain management challenges, and other external factors make the product family design problems challenging to formulate. Such dependencies are rarely addressed in current studies. If such parameters are also considered as part of the problem formulation, the provided design solutions can be pursued more confidently. Among other dynamic factors, the simultaneous and multiple effects of commonality and modularity on different members of supply chain can be addressed, which is worth to be incorporated into future studies (Fixson 2007).

*Standardization of Terminology.* While numerous modeling approaches have been developed for integrating information and knowledge about different aspects of product family design, there is still a need for clear standardization of definitions, terminology, and roles for components and elements in product families and platforms. Such a standard system enables effective reuse of design information and expedites future design improvements (Yan and Stewart 2010).

*Robust Optimization Approaches with Less Complexity.* Based on the limitations mentioned in most of the optimization studies, computational complexity is a challenging issue, which emerges from large-scale problems (Khajavirad et al. 2007). Approaches for obtaining global optimality are of interest among researchers in this field, although not many achievements have been obtained thus far, and sub-optimality is still a challenge that needs to be addressed in future research (Khajavirad and Michalek 2008; Khajavirad et al. 2009). Another thrust is to develop approaches to handle complex problems with discrete variables. While some relaxation strategies have been developed and some algorithms can handle discontinuities, most problems with similar structure (i.e., mixed variables in joint platform problems) suffer from lack of efficient and effective optimization methods. Also, improving the design space exploration techniques is fundamental for improving the current state of design optimization, as coupling the design space

exploration with the optimization techniques is shown to be essential for increasing the effectiveness of approaches to support PFD. Eventually, as integration of more objectives (i.e., platform and family performance objectives and cost targets) results in more efficient solutions, development of efficient algorithms for optimizing such multi-objective high-dimensional problems is a remarkable need in this area.

*Manufacturing and Resource Constraint Consideration.* Little attention has been paid to constraints related to resource utilization, flexible manufacturing, reconfigurability, and manufacturability (Jiao et al. 2007c, 2007d); however, PFD involves all of the stages from marketing studies to supply chain management. Particularly, very few resource constrained studies have been done, which could result in significant increase of the efficiency of design options (Kimura and Nielsen 2005).

*Global Platform Development.* Nowhere are these manufacturing and supply chain issues more prevalent than in companies that are competing globally. Numerous companies are struggling with the decision of whether to create a global platform that can be customized for regional markets around the world or regional platforms that can be more easily tailored to local markets. While user variation, competition, and government regulations and standards drive many of these decisions, subtle differences in cultures, brand perceptions, and market penetration must also be taken into account given the ramifications of one's global platform development. In some industries, the value of "Made in the USA" carries extra weight, whereas in others, a low price point is the key to market entry, yet the last thing a company wants is to cannibalize its own products by creating, for example, low-cost competitors in one market that migrate back to their high-end domestic market (Simpson et al. 2006). One example of attempt for designing global family can be seen in Fujita et al. (2012). Also, there are good lessons to be learned from the automotive industry (Miller 1999; Hodges 2004), for example, but the global market affects everyone now.

## 1.7 Summary

In this chapter, recent achievements and research activities in product family design and platform-based product development have been reviewed, emphasizing advancements that have occurred since the publication of the initial version of *Product Platform and Product Family Design: Methods and Applications* (Simpson et al. 2005). Among them, development of algorithms for handling large-scale platform selection and optimization problems, attention toward more customer-oriented designs and market-driven studies, advancements in incorporating past design information and knowledge into future improvements, and increased attention to diversity as a beneficial issue with the same importance as commonality can be addressed. However, there are still more rooms for improvement in all of these areas and also in other less tackled ones. Issues such as uncertainty considerations,

more factors' involvement into design decisions, complexity tackling in large-scale optimization problems, large design space exploration, flexibility and reconfigurability, product postponement, and supply chain issues for families are examples of areas which can be further improved in future studies about product family design and platform development.

## References

- Abdi MR (2012) Product family formation and selection for reconfigurability using analytical network process. *Int J Prod Res* 50(17):4908–4921
- Agard B, Penz B (2009) A simulated annealing method based on a clustering approach to determine bills of materials for a large product family. *Int J Prod Econ* 117(2):389–401. doi:[10.1016/j.ijpe.2008.12.004](https://doi.org/10.1016/j.ijpe.2008.12.004)
- Agard B, Cheung B, Chunda CD (2006) Selection of a module stock composition using genetic algorithm. In: 12th IFAC symposium on information control problems in manufacturing, Saint-Étienne, 17–19 May 2006
- Agrawal T, Sao A, Fernandes KJ, Tiwari MK, Kim DY (2012) A hybrid model of component sharing and platform modularity for optimal product family design. *Int J Prod Res*: 1–12. doi:[10.1080/00207543.2012.663106](https://doi.org/10.1080/00207543.2012.663106)
- Alizon F, Shooter SB, Thevenot HJ (2006) Design structure matrix flow for improving identification and specification of modules. In: Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference, Philadelphia, PA, 10–13 Sept 2006
- Alizon F, Khadke K, Thevenot HJ, Gershenson JK, Marion TJ, Shooter SB, Simpson TW (2007) Frameworks for product family design and development. *Concurr Eng* 15:187–199. doi:[10.1177/1063293X07079326](https://doi.org/10.1177/1063293X07079326)
- Alizon F, Shooter SB, Simpson TW (2009a) Assessing and improving commonality and diversity within a product family. *Res Eng Des* 20(4):241–253. doi:[10.1007/s00163-009-0066-5](https://doi.org/10.1007/s00163-009-0066-5)
- Alizon F, Shooter SB, Simpson TW (2009b) Henry Ford and the Model T: lessons for product platforming and mass customization. *Des Stud* 30(5):588–605. doi:[10.1016/j.destud.2009.03.003](https://doi.org/10.1016/j.destud.2009.03.003)
- Alizon F, Shooter SB, Simpson TW (2010) Recommending a platform leveraging strategy based on the homogeneous or heterogeneous nature of a product line. *J Eng Des* 21(1):93–110
- Allada V, Choudhury AK, Pakala PK, Simpson, TW, Scott MJ, Valliyappan S (2006) Product platform problem taxonomy: classification and identification of benchmark problems. In: Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference, Philadelphia, PA, 10–13 Sept 2006
- Anzanello MJ, Fogliatto FS (2011) Selecting the best clustering variables for grouping mass-customized products involving workers' learning. *Int J Prod Econ* 130(2):268–276
- Arciniegas AJR, Kim HM (2011) Optimal component sharing in a product family by simultaneous consideration of minimum description length and impact metric. *Eng Optim* 43(2):175–192
- Arciniegas AJR, Kim HM (2012) Incorporating security considerations into optimal product architecture and component sharing decision in product family design. *Eng Optim* 44(1):55–74
- Barajas M, Agard B (2009) The use of fuzzy logic in product family development: literature review and opportunities. Centre interuniversitaire de recherche, CIRRELT report, CIRRELT-2009-31
- Baxter D, Gao J, Case K, Harding J, Young B, Cochrane S, Dani S (2007) An engineering design knowledge reuse methodology using process modelling. *Res Eng Des* 18(1):37–48. doi:[10.1007/s00163-007-0028-8](https://doi.org/10.1007/s00163-007-0028-8)

- Belloni A, Freund R, Selove M, Simester D (2008) Optimizing product line designs: efficient methods and comparisons. *Manage Sci* 54(9):1544–1552
- Berry S, Pakes A (2007) The pure characteristics demand model. *Int Econ Rev* 48(4):1193–1225. doi:[10.1111/j.1468-2354.2007.00459.x](https://doi.org/10.1111/j.1468-2354.2007.00459.x)
- Blecker T, Abdelkafi N (2007) The development of a component commonality metric for mass customization. *IEEE Trans Eng Manage* 54(1):70–85
- Blecker T, Abdelkafi N, Kaluza B, Friedrich G (2006) Controlling variety-induced complexity in mass customisation: a key metrics-based approach. *Int J Mass Custom* 1(2–3):272–298
- Boas RC (2008) Commonality in complex product families: implications of divergence and lifecycle offsets. Ph.D. Dissertation, Massachusetts Institute of Technology
- Brière-Côté A, Rivest L, Desrochers A (2010) Adaptive generic product structure modelling for design reuse in engineer-to-order products. *Comput Ind* 61(1):53–65. doi:[10.1016/j.compind.2009.07.005](https://doi.org/10.1016/j.compind.2009.07.005)
- Callahan S (2006) Extended generic product structure: an information model for representing product families. *J Comput Inf Sci Eng* 6(3):263–275. doi:[10.1115/1.2218361](https://doi.org/10.1115/1.2218361)
- Chen AL, Martinez DH (2012) A heuristic method based on genetic algorithm for the baseline-product design. *Expert Syst Appl* 39(5):5829–5837
- Chen CB, Wang LY (2008) A modified genetic algorithm for product family optimization with platform specified by information theoretical approach. *J Shanghai Jiaotong Univ (Sci)* 13(3):304–311. doi:[10.1007/s12204-008-0304-4](https://doi.org/10.1007/s12204-008-0304-4)
- Chowdhury S, Messac A, Khire RA (2011) Comprehensive product platform planning (CP3) framework. *J Mech Des* 133:101004
- Chung SH, Lee AHI, Kang HY, Lai CW (2008) A DEA window analysis on the product family mix selection for a semiconductor fabricator. *Expert Syst Appl* 35(1–2):379–388. doi:[10.1016/j.eswa.2007.07.011](https://doi.org/10.1016/j.eswa.2007.07.011)
- Cormier P, Van Horn D, Lewis K (2009) Investigating the use of (re)configurability to reduce product family cost and mitigate performance losses. In: Proceedings of the ASME 2009 international design engineering technical conferences and computers and information in engineering conference, San Diego, CA, 30 Aug–2 Sept 2009
- Cuthrell D (1996) Product architecture. In: Rosenau MD, Griffin A, Castello GA, Anschuetz NF (eds) *The PDMA handbook of new product development*. Wiley, New York, NY
- Cutting-Decelle AF, Young RIM, Michel JJ, Grangel R, Le Cardinal J, Bourey JP (2007) ISO 15531 MANDATE: a product-process-resource based approach for managing modularity in production management. *Concurr Eng* 15(2):217–235
- Dai Z, Scott MJ (2006) Effective product family design using preference aggregation. *J Mech Des* 128(4):659–667. doi:[10.1115/1.2197835](https://doi.org/10.1115/1.2197835)
- Dai Z, Scott MJ (2007) Product platform design through sensitivity analysis and cluster analysis. *J Intell Manuf* 18(1):97–113. doi:[10.1007/s10845-007-0011-2](https://doi.org/10.1007/s10845-007-0011-2)
- Das K (2011) Integrating effective flexibility measures into a strategic supply chain planning model. *Eur J Oper Res* 211(1):170–183
- Doraszelski U, Draganska M (2006) Market segmentation strategies of multiproduct firms. *J Ind Econ* 54(1):125–149. doi:[10.1111/j.1467-6427.2006.00278.x](https://doi.org/10.1111/j.1467-6427.2006.00278.x)
- Emmatty FJ, Sarmah SP (2012) Modular product development through platform-based design and DFMA. *J Eng Des* 23(9):696–714
- Fan BB, Qi GN (2007) Modeling of production family structure and module analysis method based on complex network. *Chinese J Mech Eng* 43(3):187–192
- Farrell RS, Simpson TW (2008) A method to improve platform leveraging in a market segmentation grid for an existing product line. *J Mech Des* 130(3):031403–031411. doi:[10.1115/1.2829889](https://doi.org/10.1115/1.2829889)
- Fellini R, Kokkolaras M, Michelena N, Papalambros P, Perez-Duarte A, Saitou K, Fenyés P (2004) A sensitivity-based commonality strategy for family products of mild variation, with application to automotive body structures. *Struct Multidiscip Optim* 27(1):89–96. doi:[10.1007/s00158-003-0356-x](https://doi.org/10.1007/s00158-003-0356-x)

- Fellini R, Kokkolaras M, Papalambros P, Perez-Duarte A (2005) Platform selection under performance bounds in optimal design of product families. *J Mech Des* 127(4):524–535. doi:[10.1115/1.1899176](https://doi.org/10.1115/1.1899176)
- Fellini R, Kokkolaras M, Papalambros P (2006) Quantitative platform selection in optimal design of product families, with application to automotive engine design. *J Eng Des* 17(5):429–446
- Ferguson S, Lewis K (2008) Investigating the interaction between reconfigurability and system mass using multidisciplinary design optimization. In: 49th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference, Schaumburg, IL, 7–10 Apr 2008
- Ferguson S, Lewis K, Kasprzak E (2007a) Design and optimization of reconfigurable vehicle platforms. In: 48th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference, Honolulu, HI, 23–26 Apr 2007
- Ferguson S, Lewis K, de Weck O, Siddiqi A (2007b) Flexible and reconfigurable systems: nomenclature and review. In: Proceedings of the ASME 2007 international design engineering technical conferences and computers and information in engineering conference, Las Vegas, NV, 4–7 Sept 2007
- Ferguson S, Kasprzak E, Lewis K (2009) Designing a family of reconfigurable vehicles using multilevel multidisciplinary design optimization. *Struct Multidiscip Optim* 39(2):171–186. doi:[10.1007/s00158-008-0319-3](https://doi.org/10.1007/s00158-008-0319-3)
- Ferguson S, Olewnik A, Cormier P (2011) Exploring marketing to engineering information mapping in mass customization: a presentation of ideas, challenges and resulting questions. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Fixson SK (2007) Modularity and commonality research: past developments and future opportunities. *Concurr Eng* 15(2):85–111
- Freeman D, Lim D, Garcia E, Mavris D (2011) Identification of product family platforms using pattern recognition. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Fujita K (2002) Product variety optimization under modular architecture. *Comput Aided Des* 34(12):953–965. doi:[10.1016/s0010-4485\(01\)00149-x](https://doi.org/10.1016/s0010-4485(01)00149-x)
- Fujita K, Amaya H, Akai R (2011) Global product family design: a mathematical model for simultaneous decision of module commonalization and supply chain configuration. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Fujita K, Amaya H, Akai R (2012) Mathematical model for simultaneous design of module commonalization and supply chain configuration toward global product family. *J Intell Manuf*: 1–14. doi:[10.1007/s10845-012-0641-x](https://doi.org/10.1007/s10845-012-0641-x)
- Gao F, Xiao G, Simpson TW (2009) Module-scale-based product platform planning. *Res Eng Des* 20(2):129–141. doi:[10.1007/s00163-008-0061-2](https://doi.org/10.1007/s00163-008-0061-2)
- Georgiadis P, Athanasiou E (2010) The impact of two-product joint lifecycles on capacity planning of remanufacturing networks. *Eur J Oper Res* 202(2):420–433
- Helo P, Xu Q, Kristianto, Jiao RJ (2009) Product family design and logistics decision support system. In: IEEE 16th international conference on industrial engineering and engineering management, Beijing, 21–23 Oct 2009
- Hilletofth P, Ericsson D, Lumsden K (2010) Coordinating new product development and supply chain management. *Int J Value Chain Manage* 4(1):170–192
- Hisrich RD, Peters MP (1991) Marketing decisions for new and mature products. Prentice-Hall, Englewood Cliffs, NJ
- Hodges P (2004) Issues in automotive product platform strategies. In: 2004 SAE world congress and exhibition conference, Detroit, MI, 8 Mar 2004
- Höfner P, Khedri R, Möller B (2011) An algebra of product families. *Softw Syst Model* 10(2):161–182. doi:[10.1007/s10270-009-0127-2](https://doi.org/10.1007/s10270-009-0127-2)

- Höltkä-Otto K, de Weck O (2007) Degree of modularity in engineering systems and products with technical and business constraints. *Concurr Eng* 15(2):113–126. doi:[10.1177/1063293x07078931](https://doi.org/10.1177/1063293x07078931)
- Hoogeweegen MR, van Liere DW, Vervest PHM, van der Meijden LH, de Lepper I (2006) Strategizing for mass customization by playing the business networking game. *Decis Support Syst* 42(3):1402–1412. doi:[10.1016/j.dss.2005.11.007](https://doi.org/10.1016/j.dss.2005.11.007)
- Hou S, He L, Xie H, Liu Y (2011) Research on product family modeling method for design resource reuse. *Comput Intell Syst* 233:337–344
- Huang GQ, Li L, Chen X (2007) A tandem evolutionary algorithm for platform product customization. *J Comput Inf Sci Eng* 7(2):151–159
- Jariri F, Zegordi S (2008) Quality function deployment planning for platform design. *Int J Adv Manuf Technol* 36(5):419–430. doi:[10.1007/s00170-006-0853-3](https://doi.org/10.1007/s00170-006-0853-3)
- Jiang P, Zhao X, Yang, B, Zhao L, Tan R (2007) The product family design based on axiomatic design. In: 2007 I.E. international conference on industrial engineering and engineering management conference, Singapore, 2–5 Dec 2007
- Jiao J (2012) Product platform flexibility planning by hybrid real options analysis. *IIE Trans* 44(6):431–445
- Jiao J, Zhang Y (2005) Product portfolio identification based on association rule mining. *Comput Aided Des* 37(2):149–172. doi:[10.1016/j.cad.2004.05.006](https://doi.org/10.1016/j.cad.2004.05.006)
- Jiao J, Kumar A, Lim C (2006) Flexibility valuation of product family architecture: a real-option approach. *Int J Adv Manuf Technol* 30(1):1–9. doi:[10.1007/s00170-005-0020-2](https://doi.org/10.1007/s00170-005-0020-2)
- Jiao H, Zhang LF, Pokharel S (2007a) Process platform planning for variety coordination from design to production in mass customization manufacturing. *IEEE Trans Eng Manage* 54(1):112–129. doi:[10.1109/tem.2006.889071](https://doi.org/10.1109/tem.2006.889071)
- Jiao J, Gershenson JK, Michalek J (2007b) Managing modularity and commonality in product and process development. *Concurr Eng Res Appl* 15(2):81–83
- Jiao J, Simpson TW, Siddique Z (2007c) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18(1):5–29. doi:[10.1007/s10845-007-0003-2](https://doi.org/10.1007/s10845-007-0003-2)
- Jiao J, Zhang L, Pokharel S, He Z (2007d) Identifying generic routings for product families based on text mining and tree matching. *Decis Support Syst* 43(3):866–883. doi:[10.1016/j.dss.2007.01.001](https://doi.org/10.1016/j.dss.2007.01.001)
- Jiao J, Zhang Y, Wang Y (2007e) A generic genetic algorithm for product family design. *J Intell Manuf* 18(2):233–247. doi:[10.1007/s10845-007-0019-7](https://doi.org/10.1007/s10845-007-0019-7)
- Jin M, Chen R (2008) The platform configuration for product family production. In: 4th International conference on wireless communications, networking and mobile computing, Dalian, 19–21 Sep 2008
- Johnson MD, Kirchain RE (2009) Quantifying the effects of product family decisions on material selection: a process-based costing approach. *Int J Prod Econ* 120(2):653–668
- Johnson MD, Kirchain RE (2010) Developing and assessing commonality metrics for product families: a process-based cost-modeling approach. *IEEE Trans Eng Manage* 57(4):634–648
- Kalyanasundaram V, Lewis K (2011) A function based approach for product integration. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Kazemzadeh R, Behzadian M, Aghdasi M, Albadvi A (2009) Integration of marketing research techniques into house of quality and product family design. *Int J Adv Manuf Technol* 41(9):1019–1033. doi:[10.1007/s00170-008-1533-2](https://doi.org/10.1007/s00170-008-1533-2)
- Khajavirad A, Michalek J (2007) A single-stage gradient-based approach for solving the joint product family platform selection and design problem using decomposition. In: Proceedings of the ASME 2007 international design engineering technical conferences and computers and information in engineering conference, Las Vegas, NV, 4–7 Sept 2007

- Khajavirad A, Michalek J (2008) A decomposed gradient-based approach for generalized platform selection and variant design in product family optimization. *J Mech Des* 130(7):071101–071108. doi:[10.1115/1.2918906](https://doi.org/10.1115/1.2918906)
- Khajavirad A, Michalek J (2009) A deterministic lagrangian-based global optimization approach for quasiseparable nonconvex mixed-integer nonlinear programs. *J Mech Des* 131(5):051009-1–051009-8. doi:[10.1115/1.3087559](https://doi.org/10.1115/1.3087559)
- Khajavirad A, Michalek J, Simpson TW (2007) A decomposed genetic algorithm for solving the joint product family optimization problem. In: 3rd AIAA multidisciplinary design optimization specialists conference, Honolulu, HI, 23–26 Apr 2007
- Khajavirad A, Michalek J, Simpson TW (2009) An efficient decomposed multiobjective genetic algorithm for solving the joint product platform selection and product family design problem with generalized commonality. *Struct Multidiscip Optim* 39(2):187–201. doi:[10.1007/s00158-008-0321-9](https://doi.org/10.1007/s00158-008-0321-9)
- Khalaf REH, Agard B, Penz B (2008) Greedy heuristics for determining a product family bill of materials. In: 38th International conference on computers and industrial engineering, Beijing, 31 Oct–2 Nov 2008
- Khalaf REH, Agard B, Penz B (2009) Joint design of product family and supply chain: between diversity and standardization. Centre Interuniversitaire de Recherche, CIRRELT report, CIRRELT-2009-32
- Khalaf REH, Agard B, Penz B (2011) Simultaneous design of a product family and its related supply chain using a Tabu Search algorithm. *Int J Prod Res* 49(19):5637–5656
- Khire R, Messac A (2008) Selection-integrated optimization (SIO) methodology for optimal design of adaptive systems. *J Mech Des* 130(10):101401–101413. doi:[10.1115/1.2965365](https://doi.org/10.1115/1.2965365)
- Khire R, Messac A, Simpson TW (2006) Optimal design of product families using selection-integrated optimization (SIO) methodology. In: 11th AIAA/ISSMO symposium on multidisciplinary analysis and optimization, Portsmouth, VA, 6–8 Sep 2006
- Khire R, Wang J, Bailey T, Lin Y, Simpson TW (2008) Product family commonality selection through interactive visualization. In: Proceedings of the ASME 2008 international design engineering technical conferences and computers and information in engineering conference, New York, NY, 3–6 Aug 2008
- Kim KJ, Lee DU, Lee MS (2006) Determining product platform elements for mass customisation. *Int J Prod Qual Manage* 1(1–2):168–182
- Kimura F, Nielsen J (2005) A design method for product family under manufacturing resource constraints. *CIRP Ann Manuf Technol* 54(1):139–142. doi:[10.1016/s0007-8506\(07\)60068-7](https://doi.org/10.1016/s0007-8506(07)60068-7)
- Kumar D, Chen W, Simpson TW (2006) A market-driven approach to the design of platform-based product families. In: 11th AIAA/ISSMO symposium on multidisciplinary analysis and optimization, Portsmouth, VA, 6–8 Sep 2006
- Kumar D, Chen W, Simpson TW (2009) A market-driven approach to product family design. *Int J Prod Res* 47(1):71–104
- Kwak M, Kim HM (2011) Assessing product family design from an end-of-life perspective. *Eng Optim* 43(3):233–255
- Lamothe J, Hadj-Hamou K, Aldanondo M (2006) An optimization model for selecting a product family and designing its supply chain. *Eur J Oper Res* 169(3):1030–1047. doi:[10.1016/j.ejor.2005.02.007](https://doi.org/10.1016/j.ejor.2005.02.007)
- Li H, Azarm S (2000) Product design selection under uncertainty and with competitive advantage. *J Mech Des* 122(4):411–418
- Li H, Azarm S (2002) An approach for product line design selection under uncertainty and competition. *J Mech Des* 124(3):385–392
- Li L, Huang GQ (2009) Multiobjective evolutionary optimisation for adaptive product family design. *Int J Comput Integr Manuf* 22(4):299–314. doi:[10.1080/09511920802014920](https://doi.org/10.1080/09511920802014920)
- Li H, Liu W, Liu D (2006) The extended application quality function deployment in mass customization. *Sci Technol Manage Res* 6:227–229



- Li L, Huang GQ, Newman ST (2007) Interweaving genetic programming and genetic algorithm for structural and parametric optimization in adaptive platform product customization. *Robot Comput Integr Manuf* 23(6):650–658. doi:[10.1016/j.rcim.2007.02.014](https://doi.org/10.1016/j.rcim.2007.02.014)
- Li L, Huang GQ, Newman ST (2008) A cooperative coevolutionary algorithm for design of platform-based mass customized products. *J Intell Manuf* 19(5):507–519. doi:[10.1007/s10845-008-0137-x](https://doi.org/10.1007/s10845-008-0137-x)
- Li Z, Cheng Z, Feng Y, Yang J (2012) An integrated method for flexible platform modular architecture design. *J Eng Des iFirst*: 1–20. doi:[10.1080/09544828.2012.668614](https://doi.org/10.1080/09544828.2012.668614)
- Liang WY, Huang CC (2002) The agent-based collaboration information system of product development. *Int J Inf Manage* 22(3):211–224. doi:[10.1016/s0268-4012\(02\)00006-3](https://doi.org/10.1016/s0268-4012(02)00006-3)
- Lim SCJ, Liu Y, Lee WB (2011) A platform selection approach based on product family ontology modeling. In: *Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference*, Washington, DC, 28–31 Aug 2011
- Liu H, Özer Ö (2009) Managing a product family under stochastic technological changes. *Int J Prod Econ* 122(2):567–580. doi:[10.1016/j.ijpe.2009.06.039](https://doi.org/10.1016/j.ijpe.2009.06.039)
- Liu F, Qi G (2006) Research on evolving rule of part relation network of product family and its application. In: *International technology and innovation conference*, London, 6–7 Nov 2006
- Liu S, Tang Y, Luo S (2009) A study of product family design DNA based on product style. In: *IEEE 10th international conference on computer-aided industrial design and conceptual design*, Wenzhou, 26–29 Nov 2009
- Liu Z, Wong YS, Lee KS (2011) A manufacturing-oriented approach for multi-platforming product family design with modified genetic algorithm. *J Intell Manuf* 22(6):891–907
- Luo L (2011) Product line design for consumer durables: an integrated marketing and engineering approach. *J Market Res* 48(1):128–139. doi:[10.1509/jmkr.48.1.128](https://doi.org/10.1509/jmkr.48.1.128)
- Luo X, Tang J, Wang D (2008) Optimization of scalable product platform using quality function deployment. In: *2008 Chinese control and decision conference*, Yantai, China, 2–4 July 2008
- Luo X, Tang J, Kwong CK (2009) A product platform optimization method based on QFD. In: *IEEE 16th international conference on industrial engineering and engineering management*, Beijing, 21–23 Oct 2009
- Luo X, Kwong C, Tang J, Deng S, Gong J (2011) Integrating supplier selection in optimal product family design. *Int J Prod Res* 49(14):4195–4222
- Ma Q, Tan R, Jiang P, Yao B, Hui X (2011) Flexible product platform based on design parameters. *Building innovation pipelines through computer-aided innovation*, vol 355. Springer, Heidelberg, pp 7–15
- Marion TJ, Thevenot HJ, Simpson TW (2007) A cost-based methodology for evaluating product platform commonality sourcing decisions with two examples. *Int J Prod Res* 45(22):5285–5308
- Martin MV, Ishii K (1996) Design for variety: a methodology for understanding the costs of product proliferation. In: *Proceedings of the ASME 1996 international design engineering technical conferences and computers and information in engineering conference*, Irvine, CA, 18–22 Aug 1996
- Martin MV and Ishii K (1997) Design for variety: development of complexity indices and design charts. In: *Proceedings of the ASME 1997 international design engineering technical conferences and computers and information in engineering conference*, Sacramento, CA, 14–17 Sept 1997
- Mehrabi MG, Ulsoy AG, Koren Y (2000) Reconfigurable manufacturing systems: key to future manufacturing. *J Intell Manuf* 11(4):403–419. doi:[10.1023/a:1008930403506](https://doi.org/10.1023/a:1008930403506)
- Meng X, Jiang Z, Huang GQ (2007) On the module identification for product family development. *Int J Adv Manuf Technol* 35(1):26–40. doi:[10.1007/s00170-006-0712-2](https://doi.org/10.1007/s00170-006-0712-2)
- Meyer M, Lehnerd AP (1997) *The power of product platform—building value and cost leadership*. Free Press, New York, NY



- Michalek J, Feinberg FM, Papalambros P (2005) Linking marketing and engineering product design decisions via analytical target cascading. *J Prod Innovat Manage* 22(1):42–62
- Michalek J, Ceryan O, Papalambros P, Koren Y (2006) Balancing marketing and manufacturing objectives in product line design. *J Mech Des* 128(6):1196–1204. doi:[10.1115/1.2336252](https://doi.org/10.1115/1.2336252)
- Michalek J, Feinberg FM, Adiguzel F, Ebbes P, Papalambros P (2007) Realizable product line design optimization: coordinating marketing and engineering models via analytical target cascading. Working paper. University of Michigan, Ann Arbor, MI
- Michalek J, Ebbes P, Adiguzel F, Feinberg FM, Papalambros P (2011) Enhancing marketing with engineering: optimal product line design for heterogeneous markets. *Int J Res Market* 28(1):1–12. doi:[10.1016/j.ijresmar.2010.08.001](https://doi.org/10.1016/j.ijresmar.2010.08.001)
- Miller S (1999) VW sows confusion with common pattern for models that Investors worry profits may suffer as lines compete. *Wall Street Journal*, New York: A25
- Montano RP (2011) Platform project management: optimizing product development by actively managing commonality. MSc Thesis, Massachusetts Institute of Technology
- Moon SK, Kumara SRT, Simpson TW (2006a) Data mining and fuzzy clustering to support product family design. In: Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference, Philadelphia, PA, 10–13 Sept 2006
- Moon SK, Kumara SRT, Simpson TW (2006b) A multi-agent system for modular platform design in a dynamic electronic market environment. In: Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference, Philadelphia, PA, 10–13 Sept 2006
- Moon SK, Sim J, Shu J, Simpson TW (2007) Strategic module sharing for customized service family design using a Bayesian game. In: IEEE international conference on service operations and logistics, and informatics, Philadelphia, PA, 27–29 Aug 2007
- Moon SK, Park J, Simpson TW, Kumara SRT (2008) A dynamic multiagent system based on a negotiation mechanism for product family design. *IEEE Trans Autom Sci Eng* 5(2):234–244
- Moon SK, Simpson TW, Kumara S (2010) A methodology for knowledge discovery to support product family design. *Ann Oper Res* 174(1):201–218. doi:[10.1007/s10479-008-0349-7](https://doi.org/10.1007/s10479-008-0349-7)
- Moon SK, Park J, Simpson TW (2011) Platform strategy for product family design using particle swarm optimization. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Morrise JS, Lewis K, Mattson CA, Magleby SP (2011) A method for designing collaborative products with application to poverty alleviation. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Mula J, Peidro D, Poler R (2010) The effectiveness of a fuzzy mathematical programming approach for supply chain production planning with fuzzy demand. *Int J Prod Econ* 128(1):136–143
- Nanda J, Thevenot HJ, Simpson TW, Stone RB, Bohm M, Shooter SB, Baader F, Calvanese D, McGuinness D, Nardi D (2007) Product family design knowledge representation, aggregation, reuse, and analysis. *Artif Intell Eng Des Anal Manuf* 21(2):173
- Nomaguchi Y, Taguchi T, Fujita K (2006) Knowledge model for managing product variety and its reflective design process. In: Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference, Philadelphia, PA, 10–13 Sept 2006
- Olewnik A, Lewis K (2006) A decision support framework for flexible system design. *J Eng Des* 17(1):75–97. doi:[10.1080/09544820500274019](https://doi.org/10.1080/09544820500274019)
- Öman M, Nilsson L (2011) An improved critical constraint method for structural optimization of product families. *Struct Multidiscip Optim* 45(2):235–246. doi:[10.1007/s00158-011-0689-9](https://doi.org/10.1007/s00158-011-0689-9)
- Ong SK, Xu QL, Nee AYC (2006) Design reuse methodology for product family design. *CIRP Ann Manuf Technol* 55(1):161–164. doi:[10.1016/s0007-8506\(07\)60389-8](https://doi.org/10.1016/s0007-8506(07)60389-8)

- Ostrosi E, Fougères AJ, Ferney M, Klein D (2011) A fuzzy configuration multi-agent approach for product family modelling in conceptual design. *J Intell Manuf* 23(6):2565–2586. doi:[10.1007/s10845-011-0541-5](https://doi.org/10.1007/s10845-011-0541-5)
- Otto K, Hölttä-Otto K (2007) A multi-criteria assessment tool for screening preliminary product platform concepts. *J Intell Manuf* 18(1):59–75. doi:[10.1007/s10845-007-0004-1](https://doi.org/10.1007/s10845-007-0004-1)
- Pandey V, Thurston D (2008) Metric for disassembly and reuse decisions: formulation and validation. In: Proceedings of the ASME 2008 international design engineering technical conferences and computers and information in engineering conference, New York, NY, 3–6 Aug 2008
- Park J, Simpson TW (2008) Toward an activity-based costing system for product families and product platforms in the early stages of development. *Int J Prod Res* 46(1):99–130
- Pasche MH, Persson M, Löfsten H (2011) Effects of platforms on new product development projects. *Int J Oper Prod Manage* 31(11):1144–1163. doi:[10.1108/01443571111178475](https://doi.org/10.1108/01443571111178475)
- Perez AG, Linsey JS (2011) Identifying product scaling principles: a tool for bioinspired design and beyond. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Pine BJ (1993) *Mass customization: the new frontier in business competition*. Harvard Business School Press, Boston, MA
- Qian L (2009) Evaluate cost and time in supply chain selection under price-dependent demand for one product family. In: IEEE/INFORMS international conference on service operations, logistics and informatics, Chicago, IL, 22–24 July 2009
- Qin Y, Wei G (2009) On mapping approach of CN to FR in product family improvement. In: International conference on measuring technology and mechatronics automation, Zhangjiajie, 11–12 Apr 2009
- Qu T, Bin S, Huang GQ, Yang H (2011) Two-stage product platform development for mass customisation. *Int J Prod Res* 49(8):2197–2219
- Raffaelli R, Mengoni M, Germani M (2011) An early-stage tool to evaluate the product redesign impact. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Rojas AJ, Esterman JM (2008) A measure of impact for platform changes. In: Proceedings of the ASME 2008 international design engineering technical conferences and computers and information in engineering conference, New York, NY, 3–6 Aug 2008
- Salhieh SEM (2007) A methodology to redesign heterogeneous product portfolios as homogeneous product families. *Comput Aided Des* 39(12):1065–1074. doi:[10.1016/j.cad.2007.07.005](https://doi.org/10.1016/j.cad.2007.07.005)
- Sandborn P, Prabhakar V, Eriksson B (2008) The application of product platform design to the reuse of electronic components subject to long-term supply chain disruptions. In: Proceedings of the ASME 2008 international design engineering technical conferences and computers and information in engineering conference, New York, NY, 3–6 Aug 2008
- Schönsleben P (2012) Methods and tools that support a fast and efficient design-to-order process for parameterized product families. *CIRP Ann Manuf Technol* 61(1):179–182
- Scott MJ, Arenillas J, Simpson TW, Valliyappan S, Allada V (2006) Towards a suite of problems for comparison of product platform design methods: a proposed classification. In: Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference, Philadelphia, PA, 10–13 Sept 2006
- Seol H, Kim C, Lee C, Park Y (2007) Design process modularization: concept and algorithm. *Concurr Eng* 15(2):175–186. doi:[10.1177/1063293x07079321](https://doi.org/10.1177/1063293x07079321)
- Shahzad KM, Hadj-Hamou K (2012) Integrated supply chain and product family architecture under highly customized demand. *J Intell Manuf*. doi:[10.1007/s10845-012-0630-0](https://doi.org/10.1007/s10845-012-0630-0)
- Shamsuzzoha A, Helo PT (2012) Development of modular product architecture through information management. *VINE* 42(2):172–190. doi:[10.1108/03055721211227200](https://doi.org/10.1108/03055721211227200)

- Sharman DM, Yassine AA (2007) Architectural valuation using the design structure matrix and real options theory. *Concurr Eng* 15(2):157–173
- Shiau CSN, Michalek J (2009a) Optimal product design under price competition. *J Mech Des* 131(7):071003–071010. doi:[10.1115/1.3125886](https://doi.org/10.1115/1.3125886)
- Shiau CSN, Michalek J (2009b) Should designers worry about market systems? *J Mech Des* 131(1):011011–011019. doi:[10.1115/1.3013848](https://doi.org/10.1115/1.3013848)
- Shooter S, Evans C, Simpson TW (2007) Building a better ice scraper—a case in product platforms for the entrepreneur. *J Intell Manuf* 18(1):159–170. doi:[10.1007/s10845-007-0010-3](https://doi.org/10.1007/s10845-007-0010-3)
- Siddiqi A, de Weck O, Iagnemma K (2006) Reconfigurability in planetary surface vehicles: modeling approaches and case study. *J Brit Interp Soc* 59(12):450–460
- Siddique Z, Wilmes L (2007) An application of design space for assembly process reasoning to utilize current assembly plant resources for new product family members. *J Intell Manuf* 18(1):171–184. doi:[10.1007/s10845-007-0013-0](https://doi.org/10.1007/s10845-007-0013-0)
- Simpson TW (2005) Methods for optimizing product platforms and product families: overview and classification. In: Simpson TW, Siddique Z, Jiao J (eds) *Product platform and product family design*. Springer, New York, NY, pp p133–p156
- Simpson TW, Siddique Z, Jiao J (2005) Platform-based product family development. In: Simpson TW, Siddique Z, Jiao J (eds) *Product platform and product family design*. Springer, New York, NY, pp p1–p15
- Simpson TW, Marion T, de Weck O, Holttä-Otto K, Kokkolaras M, Shooter SB (2006) Platform-based design and development: current trends and needs in industry. In: *Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference*, Philadelphia, PA, 10–13 Sept 2006
- Simpson TW, Bobuk A, Slingerland LA, Brennan S, Logan D, Reichard K (2012) From user requirements to commonality specifications: an integrated approach to product family design. *Res Eng Des* 23(2):141–153
- Sköld M, Karlsson C (2012) Product platform replacements: challenges to managers. *Int J Oper Prod Manage* 32(6):746–766
- Steva ED, Rice EN, Marion TJ, Simpson TW, Stone RB (2006) Two methodologies for identifying product platform elements within an existing set of products. In: *Proceedings of the ASME 2006 international design engineering technical conferences and computers and information in engineering conference*, Philadelphia, PA, 10–13 Sept 2006
- Suh E, de Weck O, Chang D (2007) Flexible product platforms: framework and case study. *Res Eng Des* 18(2):67–89. doi:[10.1007/s00163-007-0032-z](https://doi.org/10.1007/s00163-007-0032-z)
- Sullivan E, Ferguson S, Donndelinger J (2011) Exploring heterogeneity of customer preference to balance commonality and market coverage. In: *Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference*, Washington, DC, 28–31 Aug 2011
- Tao J, Yu S (2012) Incorporating reuse and remanufacturing in product family planning. In: Matsumoto M, Umeda Y, Masui K, Fukushige S (eds) *Design for innovative value towards a sustainable society*. Springer, Heidelberg, pp 795–800. doi:[10.1007/978-94-007-3010-6\\_162](https://doi.org/10.1007/978-94-007-3010-6_162)
- Thevenot HJ, Simpson TW (2006) Commonality indices for product family design: a detailed comparison. *J Eng Des* 17(2):99–119. doi:[10.1080/09544820500275693](https://doi.org/10.1080/09544820500275693)
- Thevenot HJ, Simpson TW (2007a) A comprehensive metric for evaluating component commonality in a product family. *J Eng Des* 18(6):577–598
- Thevenot HJ, Simpson TW (2007b) Guidelines to minimize variation when estimating product line commonality through product family dissection. *Des Stud* 28(2):175–194. doi:[10.1016/j.destud.2006.07.004](https://doi.org/10.1016/j.destud.2006.07.004)
- Thevenot HJ, Steva ED, Okudan GE, Simpson TW (2007) A multiattribute utility theory-based method for product line selection. *J Mech Des* 129(11):1179–1184. doi:[10.1115/1.2771574](https://doi.org/10.1115/1.2771574)
- Thomas E (2012) Exploring the strategic use of platform-based planning. *Atlantic Mark J* 1(1):3

- Torstenfelt B, Klarbring A (2006) Structural optimization of modular product families with application to car space frame structures. *Struct Multidiscip Optim* 32(2):133–140. doi:[10.1007/s00158-005-0568-3](https://doi.org/10.1007/s00158-005-0568-3)
- Torstenfelt B, Klarbring A (2007) Conceptual optimal design of modular car product families using simultaneous size, shape and topology optimization. *Finite Elem Anal Des* 43 (14):1050–1061. doi:[10.1016/j.finel.2007.06.005](https://doi.org/10.1016/j.finel.2007.06.005)
- Trentin A, Salvador F, Forza C, Rungtusanatham MJ (2011) Operationalising form postponement from a decision-making perspective. *Int J Prod Res* 49(7):1977–1999
- Tucker CS, Kim HM (2007) Product family concept generation and validation through predictive decision tree data mining and multilevel optimization. In: Proceedings of the ASME 2007 international design engineering technical conferences and computers and information in engineering conference, Las Vegas, NV, 4–7 Sept 2007
- Tucker CS, Kim HM (2008) Optimal product portfolio formulation by merging predictive data mining with multilevel optimization. *J Mech Des* 130(4):041103
- Tucker CS, Kim HM, Barker DE, Zhang Y (2010) A reliefF attribute weighting and x-means clustering methodology for top-down product family optimization. *Eng Optim* 42(7):593–616
- Turner CS, Ferguson S, Donndelinger J (2011) Exploring heterogeneity of customer preference to balance commonality and market coverage. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Van Wie M, Stone RB, Thevenot HJ, Simpson TW (2007) Examination of platform and differentiating elements in product family design. *J Intell Manuf* 18(1):77–96. doi:[10.1007/s10845-007-0005-0](https://doi.org/10.1007/s10845-007-0005-0)
- Vlachos D, Georgiadis P, Iakovou E (2007) A system dynamics model for dynamic capacity planning of remanufacturing in closed-loop supply chains. *Comput Oper Res* 34(2):367–394
- Wang Z (2010) Study on the supply chain system for the product family based on the complex network. In: 2nd International conference on e-business and information system security, Wuhan, 22–23 May 2010
- Wang Z, Ning F (2007) Study on supply chain management for product family in mass customization. In: IEEE international conference automation and logistics, Jinan, 18–21 Aug 2007
- Wang Z, Wang C (2009) Research on the collaborative allocation of general modules in product-family-oriented supply chain in mass customization. In: International conference on e-business and information system security, Wuhan, 23–24 May 2009
- Wang L, Song B, Li X, Ng WK (2007a) A product family based life cycle cost model for part variety and change analysis. In: International conference on engineering design, Paris, 28–31 Aug 2007
- Wang WD, Qin XS, Yan XT, Tong SR, Sha QY (2007b) Developing a systematic method for constructing the function platform of product family. In: 2007 I.E. international conference on industrial engineering and engineering management, Singapore, 2–5 Dec 2007, pp 60–64
- Wang ZH, Gu XJ, Qi GN (2007c) Research on supply chain optimization based on generic bills of materials of product family. In: 2007 I.E. international conference on automation and logistics, Jinan, 18–21 Aug 2007
- Wäppling D, Feng X, Andersson H, Pettersson M, Lunden B, Weström J (2011) Simultaneous requirement and design optimization of an industrial robot family using multi-objective optimization. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, Washington, DC, 28–31 Aug 2011
- Williams CB, Allen JK, Rosen DW, Mistree F (2007) Designing platforms for customizable products and processes in markets of non-uniform demand. *Concurr Eng* 15(2):201–216. doi:[10.1177/1063293x07079328](https://doi.org/10.1177/1063293x07079328)
- Williams N, Azarm S, Kannan P (2010) Multicategory design of bundled products for retail channels under uncertainty and competition. *J Mech Des* 132(3):031003.1–031003.10

- Xu Y, Xiong L, Wang Y (2006) The method of module partition for product family structure with applications. In: IEEE international conference on service operations and logistics, and informatics, Shanghai, 21–23 June 2006
- Xu QL, Ong SK, Nee AYC (2007) Evaluation of product performance in product family design re-use. *Int J Prod Res* 45(18):4119–4141
- Xu X, Gao L, Fang S (2008) Product family architecture evolution based on technology evolution theory of TRIZ. In: 7th world congress on intelligent control and automation, Chongqing, 25–27 June 2008
- Yan XT, Stewart B (2010) Developing modular product family using GeMoCURE within an SME. *Int J Manuf Res* 5(4):449–463
- Yan XT, Stewart B, Wang W, Tramsheck R, Liggat J, Duffy AHB, Whitfield I (2007) Developing and applying an integrated modular design methodology within a SME. In: International conference on engineering design, Paris, 28–31 Aug 2007
- Yang Y, Wang X (2009) A modeling approach for correlation's evolution of modular product family. *Computer-Aided Industrial Design and Conceptual Design*, IEEE 10th International Conference on CAID & CD 2009, Wenzhou, 26–29 Nov 2009
- Ye X, Gershenson JK (2008) Attribute-based clustering methodology for product family design. *J Eng Des* 19(6):571–586
- Ye X, Thevenot HJ, Alizon F, Gershenson JK, Khadke K, Simpson TW, Shooter SB (2009) Using product family evaluation graphs in product family design. *Int J Prod Res* 47(13):3559–3585
- You F, Grossmann IE, Wassick JM (2010) Multisite capacity, production, and distribution planning with reactor modifications: MILP model, bilevel decomposition algorithm versus Lagrangean decomposition scheme. *Ind Eng Chem Res* 50(9):4831–4849
- Yu J, Cai M (2009) Product master structure for product family. In: International conference on management and service science, Wuhan, 16–18 Sept 2009
- Yu TL, Yassine A, Goldberg D (2007) An information theoretic method for developing modular architectures using genetic algorithms. *Res Eng Des* 18(2):91–109. doi:[10.1007/s00163-007-0030-1](https://doi.org/10.1007/s00163-007-0030-1)
- Zacharias N, Yassine A (2007) Platform investment decisions in product family design. In: Proceedings of the ASME 2007 international design engineering technical conferences and computers and information in engineering conference, Las Vegas, NV, 4–7 Sept 2007
- Zacharias N, Yassine A (2008) Optimal platform investment for product family design. *J Intell Manuf* 19(2):131–148. doi:[10.1007/s10845-007-0069-x](https://doi.org/10.1007/s10845-007-0069-x)
- Zha XF, Sriram RD (2006) Platform-based product design and development: a knowledge-intensive support approach. *Knowl Based Syst* 19(7):524–543. doi:[10.1016/j.knosys.2006.04.004](https://doi.org/10.1016/j.knosys.2006.04.004)
- Zhang W, Fan Y (2006) Research on domain-based generic product family architecture modeling. In: The 6th world congress on intelligent control and automation, Dalian, 21–23 June 2006
- Zhang H, Zhao W, Li G, Tan R (2006a) A process model of product platform building for the products with the same function structure. In: Wang K, Kovacs G, Wozny M, Fang M (eds) *Knowledge enterprise: intelligent strategies in product design, manufacturing, and management*, vol 207. Springer, Boston, MA, pp 1002–1009
- Zhang L, Jiao J, Helo P (2006b) Integrated product and process family data modeling for product lifecycle management. In: IEEE international conference on industrial informatics, Singapore, 16–18 Aug 2006
- Zhang W, Tor S, Britton G (2006c) Managing modularity in product family design with functional modeling. *Int J Adv Manuf Technol* 30(7):579–588. doi:[10.1007/s00170-005-0112-z](https://doi.org/10.1007/s00170-005-0112-z)
- Zhang Y, Jiao JR, Ma Y (2007) Market segmentation for product family positioning based on fuzzy clustering. *J Eng Des* 18(3):227–241
- Zhao Y, Zhang M, Su N, Chen J (2010) Product family extension configuration design: the theory and method. In: The 2nd international conference on computer and automation engineering, Singapore, 26–28 Feb 2010
- Zhong DQ, Zhong WJ (2006) Research on optimal number of suppliers based on co-opetition. *J Manage Sci China* 6:52–57

**Part I**  
**Platform Planning and Strategy**

# Chapter 2

## Crafting Platform Strategy Based on Anticipated Benefits and Costs

Bruce G. Cameron and Edward F. Crawley

**Abstract** In this chapter, we introduce the benefits and penalties of commonality (both to the customer and the manufacturer), emphasizing the need for anticipation of divergence when estimating benefits. We highlight the importance of mapping commonality strategy to the financial benefits, with a view to creating long-term competitive advantage for the firm.

### 2.1 Introduction

Platforming, the sharing of products or processes across products, has become an important means of cost-sharing across industrial products. Examples include Volkswagen’s MQB platform (including VW Golf, Audi A3, and Seat Octavia) (Pander 2012), the Joint Strike Fighter program (variants for the Air Force, Marines, and Navy), and Black and Decker’s electric hand tools (Meyer and Lehnerd 1997).

The use of platform over the last three decades has grown in response to market demand for variety. Consumers have come to expect \$50, \$100, and \$150 version of a hand drill to choose from (Halman et al. 2003). Car buyers now enjoy bundled option packages (Basic, Leather, SportPlus), supported by option code sheets that could fill a book. This variety has a direct impact on the firm—for example, one automotive model can have as many as five million possible variants, when considering all of the offered options in combination (Cameron 2011). The process complexity deployed to support this market variety can threaten the organization’s survival. A recent study of wasted complexity at Proctor and Gamble identified \$3 billion in savings (Wilson and Perumal 2009).

---

B.G. Cameron (✉) • E.F. Crawley  
Massachusetts Institute of Technology, Cambridge, MA 02139, USA  
e-mail: [bcameron@alum.mit.edu](mailto:bcameron@alum.mit.edu)

Platforming is a strategy for providing variety to the market against a reduced cost base. When executed well, it can provide a vital competitive advantage to the firm. Firms have cut costs by 30 % and reduced lead times by 50 % by employing commonality (Pander 2012). The ability to bring products to market quickly and cheap can create significant first mover advantage. However, gaining this competitive advantage is not quick or cheap. The list of firms that have attempted to build platforms and failed is long. Many firms fail to reach their commonality targets—the Joint Strike Fighter has famously seen divergence from 80–90 % parts commonality down to 30–40 % parts commonality (Boas et al. 2012). A senior executive in Automotive stated his belief that learning platforming takes at least two product lifecycles.

Sharing parts does not fundamentally create competitive advantage. Commonality as a strategy is only successful insofar as it enables financial advantages, be it increased revenue or decreased cost. In fact, we will show that platforming requires significant upfront risk, in the form of large multiproduct investments and downstream risk of low product differentiation—platforms can negatively affect the firm’s brand.

We begin an examination of platform strategy by weighing the benefits and costs. We argue that the firm’s ability to achieve a competitive advantage through platforming is rooted in a meaningful strategy process, examining the investment required against the downstream savings. In this chapter, we first provide a holistic overview of the benefits. Then we examine the associated drawbacks and costs. We review the data on divergence in commonality, to understand the potential downside risk. Finally, we illustrate how the choice of commonality strategy (what to make common) should be mapped to the desired benefits to be achieved.

## 2.2 Trade-Offs in Platforming

The discussion of platforming and commonality as a strategy is perhaps best illustrated in the context of trade-offs posed by this choice of strategy, as revealed in the literature. These trade-offs arise from conserved parameters and shared efforts—examining them provides a starting point for examining cost dynamics.

In platform development, there are a number of high-level trade-offs posed at the beginning of the platform development (Otto and Hölttä-Otto 2007). The trade-offs are critically related to the main architectural parameters, such as number of variants, range of performance, sequencing of variants, and degree of commonality. In turn, the decisions about these parameters are made about the expected markets for the variants, whose relevant characteristics here are performance requirements, willingness to pay, and availability/timeliness. The market “causes” the first set of trade-offs we explore.



### ***2.2.1 Trade-Offs Caused by the Market***

Firms create multiple variants for market reasons. Customers grouped by similar pricing and performance expectations can represent submarkets, which if served individually can represent greater overall profit than producing a product which serves their average expectation. Meyer and Lehnerd (1997) originally described a process for segmenting a market using a grid tool, illustrating a number of different strategies for spreading commonality investment across a range of product prices and market segments.

These market-facing tensions have been framed in the literature as a trade between variety and commonality. Ramdas (2003) segments the market implications of variety into four categories—the dimensions of variety, the product architecture, the degree of customization, and the timing of variety. In particular, research on understanding the costs of variety forms an important counterpoint in the tension between variety and commonality (MacDuffie et al. 1996; Martin et al. 1998; Du et al. 2001; Blecker and Abdelkafi 2006). Further, the trade between closed set discrete variety (e.g., along a linear dimension of variety such as horsepower) and the potential for mass customization has been a fruitful direction of research (Alptekinoglu and Corbett 2008; Blecker and Abdelkafi 2007; Jiao and Tseng 2000; Rungtusanatham and Salvador 2008). Research has begun to unpack the underlying mechanisms which create the variety—commonality trade-off—Rungtusanatham and Salvador (2008) note that difficulties identifying latent needs and differentiation opportunities within marketing activities can lead to static offerings.

Commonality strategies architected to deliver this variety in turn create the threat of cannibalization (Sanderson and Uzumeri 1995; Kim and Chhajed 2000), where customers with higher willingness to pay can meet their performance requirements by buying the lower-performance product. Sanderson and Uzumeri (1995) describe a case in the DRAM market, illustrating how sales trajectories can show both within-platform cannibalization and generation to generation platform cannibalization. Absent detailed customer data allowing the manufacturer to bucket variant sales by segment, cannibalization can be weakly inferred from sales trajectories and product introduction timing, but the quality of the inference varies. Variants that are closely spaced are easier to platform but are at greater risk of cannibalization. One mechanism of this cannibalization is that shared components in the lowest cost variant may be subject to quality standards as applied to higher performance variants. Ulrich et al. (1998) find “for low-quality segments, brand price-premium is significantly positively correlated with the quality of the lowest quality model in the product line” (Ramdas 2003). Viewed from the other perspective, Nelson et al. (2001) describe how overdesigning lower-level variants can place acquisition and maintenance costs above the reach of some customers, thus decreasing expected platform volume and profitability.

In addition to the threats to submarkets created by platforming, there is an overall brand threat. Cook (1997) notes, “ironically GM’s market share relative to Ford only began to recede in the mid 1980s as GM’s brands—Chevrolet, Pontiac,

Oldsmobile, Buick, and Cadillac—became less distinctive through the use of common platforms and exterior stampings that reduced product differentiation” [reproduced from de Weck (2006)]. The concept of a trade-off between perceived product differentiation (and its effect on sales) and the benefits of platforming is a difficult one to measure, in that brand is influenced by many factors, and the signal from product differentiation is spread among the timings of the individual variant introductions.

The idea of flexibility of platforms is related, in that platforms can create opportunities for future variants, opportunities which are only revealed over time. The existence of a relevant platform can speed time to market, and also reduces development cost for the variant. There are existing tools for comparing flexibility’s benefits against costs. Namely, Triantis (2000), Otto et al. (2003), Jiao et al. (2006), and Rhodes (2010) have framed commonality as a real option.

Baldwin and Clark (2000) argue that modularity has been a central driver of innovation and growth at an industry level, working from deep studies in the computer industry. It is important to note that this growth did not necessarily accrue to all firms—the final external trade-off that we note is a potential threat posed by competitors entering value-creating segments of the market on top of the firm’s platform.

### ***2.2.2 Internal Trade-Offs***

Thus far, we have described the trade-offs with external influences. There are also a number of trade-offs that emerge through the development cycle. For example, firms often desire flat development budget profiles. If the concurrent development of the platform and all of its variants doesn’t fit under this flat budget, a common technique is to phase variant development. Boas (2008) describes the trade-off created between phasing development and divergence from the platform exacerbated by the offset. Cusumano and Nobeoka (1998) describe a set of strategies for phasing development (ranging from parallel to sequential), highlighting how overlapping development phases, which he titles “rapid design transfer strategy,” can strike a balance in this trade-off. Additionally, Cusumano and Nobeoka (1998) highlight how development head count time series represent a possible measurement of the phasing of development effort.

Insofar as platforms are large product development programs, they embody a whole host of constraints not specific to platforms. Personnel constraints create constraints for platforms, in that faster ramp up and ramp down times come at the expense of challenging training and quality. Existing manufacturing facilities constrain total capacity and inventory. Past capital equipment constrains current production methods as well as future capital availability (Rungtusanatham and Salvador 2008). These factors apply broadly to product development, so we do not explore in depth here—where appropriate, they are raised below in conjunction with specific platforming issues.

Work in the engineering literature has defined a variety of metrics, with a view to watching one of the key state variables, the actual level of commonality. In theory, each of the trade-offs should result in movement of an appropriately set commonality metric. For example, Thevenot and Simpson (2007) take manufacturing costs into account with a commonality metric where parts are weighted by cost, building on earlier work by Jiao and Tseng (2000).

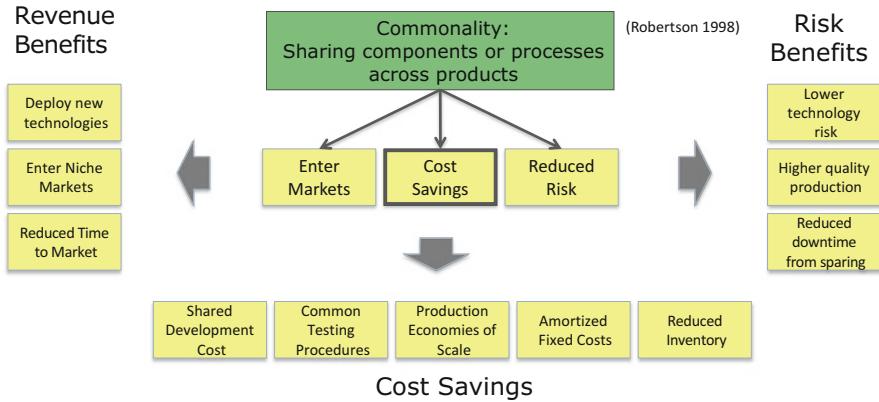
We can sum up the internal trade-offs resulting from commonality in three key criteria for commonality (Cameron 2011). Commonality strategies must be grounded in technical feasibility—a concept of a design that can be expected to span a range of performance. Commonality strategies must be financially beneficial—commonality is a means to an end. Finally, commonality must be organizationally possible—shared designs and co-investments in future products must be supported by organizational structure and process.

## 2.3 Benefits of Commonality

Much has been written on the topic of platforming and commonality, primarily stemming from seminal work by Utterback and Meyer (1993) and Robertson and Ulrich (1998), although earlier work can be found from 20 years previous (Collier 1981). These early works cited a number of benefits, such as enabling future rapid product introduction, increase model introduction rate, decreased development cost, economies of scale in manufacturing, and faster introduction of new technology into existing product lines. Since the early work on platforming, a broad body of literature has grown up around the concept of platforming, but no consensus around the list of benefits has emerged, despite several past efforts to build a list of pros and cons—see Fisher et al. (1999).

To begin, we break the benefits of commonality into three categories: (1) Revenue Benefits, (2) Cost Savings, and (3) Risk Benefits. Figure 2.1 shows examples of the tangible benefits possible in each of these categories. We delve deeper into these benefits in the remainder of this chapter.

Embedded in the notion of benefits and penalties in the management literature is the idea that managers weigh these factors when making rational decisions. As compared to the more quantitative literature on commonality, the diversity of benefits in the management literature is broad by comparison and is most likely to discuss commonality decision-making as grounded in organizational structure. As a potential frame of reference, van Maanen's organizational decision-making separates decisions into rational strategic, political, and cultural. The rational strategic frame is dominant in the management literature. However, political decisions (the embodiment of organizational power or position) are also referenced, such as in Cusumano's (1998) discussion of heavyweight program managers. Cultural decision-making is referenced in passing, such as creating a culture of reuse, but



**Fig. 2.1** Three benefits of commonality: (1) market benefits, (2) cost savings, and (3) risk benefits

has not been the subject of much descriptive work. We have found that decisions are dominantly framed under investments, as discussed in the following section.

Based on the cited literature and over 30 case studies on commonality (Wicht and Crawley 2012; Boas et al. 2012; Rhodes 2010; Cameron 2011), we have constructed a comprehensive list of commonality benefits (see Table 2.1). We have divided the benefits of commonality into five categories, roughly aligned in the order in which they occur. Cost Saving benefits are listed primarily under the phase of the product lifecycle in which they occur—Design, Manufacturing, Testing, and Operations. In addition to the traditional breakdown of a product lifecycle into Design, Manufacturing, Testing, and Operations, we have included Strategy Benefits, to explicitly recognize that some of the benefits relate more closely to Revenue Benefits and Risk Benefits than to Cost Savings.

It is important to note that not all of these benefits accrue to every platform. Additionally, we have explicitly separated reuse benefits from proactive commonality benefits. Reuse benefits in a sense exclude prior development work from the platform system boundary, in that future commonality was not intended (Unintended Commonality). Proactive commonality benefits, which comprise the majority of the table, include the initial investment and variants inside the Platform system boundary (Intentional Commonality).

The benefits of commonality from a product family planning perspective are primarily captured in the Strategy Phase. Recognizing that it is rare that the scope of product families (the platform extent, the number of variants, the performance/cost of each variant) is known entirely in advance, some of these benefits accrue due to the uncertainty in the planning phase. For example, the firm’s flexibility to enter niche markets once the platform has been defined represents an important strategic benefit (Pine 1993; Meyer and Lehnerd 1997). By contrast, within the originally forecast platform scope, platforms can help companies reduce their time to market (Clark and Fujimoto 1991; Meyer et al. 1997), as less overall design, test, and manufacturing work is required overall to bring several variants to market.

**Table 2.1** List of commonality benefits (Note: the benefits are not causal or assured, but rather the potential to achieve the benefit has been shown to exist)

Phase	Benefit	Rationale	References
Strategy	Enable faster variant time to market	The common portion of the design has already been built, so only the unique portion has to be designed	Clark and Fujimoto (1991), Meyer et al. (1997)
	Enter niche markets	Designing unforecast variants on top of the common platform enables the firm to recognize and enter markets as they appear	Pine (1993) Meyer and Lehnerd (1997), Robertson and Ulrich (1998)
	Deploy new technologies	Time and cost to deploy technologies is reduced where interfaces to the platform are identical.	Meyer (1997), Jiao et al. (2007)
	Lower technology risk	Increased investment in common technology (can also be a higher risk)	Meyer and Lehnerd (1997)
Design	Shared development cost (intended commonality)	Reduced engineering effort required for later variants	Meyer (1997), Ho and Li (1997), Johnson and Kirchain (2010)
	Reuse of already designed components and systems (unintended commonality)	Design effort does not need to be repeated	Ulrich and Ellison (1999)
Manufacture	Reuse of proven technologies	Reduces technology risk and mitigation cost	Robertson and Ulrich (1998)
	Shared tooling	Tooling cost can be spread over more products	Lehnerd (1987), Park and Simpson (2005)
	Learning curve benefits	Fewer hours/unit required	Park and Simpson (2005)
	Economies of scale in manufacturing	Enables movement to higher volume methods	Robertson and Ulrich (1998), Krishnan and Gupta (2001)
	Bulk purchasing	Discounts from suppliers for larger orders of same part	Robertson and Ulrich (1998), Simpson (2004)
	Reduced inventory Reduced quality expense Flexibility in variant volumes (for a fixed platform extent)	Lower safety stock levels due to demand aggregation Fixed quality expenses spread over larger volume Enables the firm to adjust to variant demand changes	Collier (1981), Baker et al. (1986) Sanderson and Uzumeri (1995) Suárez et al. (1991), Robertson and Ulrich (1998)

(continued)

Table 2.1 (continued)

Phase	Benefit	Rationale	References
Testing and commissioning	Reduced testing and commissioning time	Learning in test procedures for later variants	Park and Simpson (2005)
	Shared testing equipment	Testing equipment can be spread over more products	Robertson and Ulrich (1998), Park and Simpson (2005)
Operation	Reduced external testing/certification	Reuse of type certificates or regulatory approval	Rothwell and Gardiner (1990), Sabbagh (1996)
	Reduced sustaining engineering	Number of parts to be sustained is reduced	Fixson (2006)
	Decreased fixed costs from shared facilities	Sharing of facility cost across more products	Fixson (2006)
	Decreased operator training	Operator learning on common parts reduces training	Halman et al. (2003)
	Economies of scale in operations	Move to higher volume operating procedures	Halman et al. (2003)
	Bulk purchasing of consumables	Discounts from suppliers for larger order of same parts	Robertson and Ulrich (1998), Simpson (2004)
	Decreased variable costs due to more efficient logistics and sparing	Reduced inventory for operations	Collier (1981), Baker et al. (1986)
	Slower replacement rate for spares (higher quality)	Fewer spares must be purchased	Sanderson and Uzumeri (1995)
	Flexibility in operations	Ability to switch operating staff between products	Halman et al. (2003)
	Shared inspections/recurring regulatory compliance	Lower cost and less time required for regulatory compliance	Rothwell and Gardiner (1990), Sabbagh (1996)

In the design phase, commonality primarily acts to reduce the number of engineering hours required to produce a variant (Ho and Li 1997; Johnson and Kirchain 2010). Intuitively, this can be understood as engineers producing fewer unique parts. However, as seen under Costs of Commonality, common parts often take more time to design, so the effort required must be carefully sized. In addition to producing fewer parts, design hours are reduced when effort in product definition (requirements and goal setting) can be reused, when design analysis methodologies can be reapplied to slightly different parts or environments, and when challenges in the initial variant design inform design strategies for unique parts on later variants. The reduction in engineering effort is primarily measured in engineering head count or engineering hours. While these may appear to be easily applied summary measures, the realities of accounting for reduced head count on a subsequent variant as traceable to early design effort can be complex to track (Ben-Arieh and Qian 2003).

In the manufacturing phase, commonality impacts many different departments involved in coordinating manufacturing. On the physical manufacturing line, platforms can enable the firm to move to higher volume manufacturing methods, such as from operator-assisted sheet-metal bending to fully automated operations. This is typically referred to as economies of scale, in reference to the idea that higher volumes allow new capital equipment to be amortized across higher volumes (Krishnan and Gupta 2001). This should be contrasted with learning curves on the manufacturing line, the idea that the labor portion of the manufacturing cost shrinks as assemblers find more efficient ways to complete the task and reduce quality expense when the resulting efficiency causes fewer defects, particularly when the platform is designed to the higher quality variant (Desai et al. 2001). Off the physical line, the purchasing department stands to gain leverage with increasing volume of common parts, and the supply chain department can stock fewer parts, as the aggregation of demand from different products for the same common parts lowers the safety stock that needs to be carried. Fixson (2006) notes that a number of supporting costs reductions are also achieved under commonality through lower product support activities, highlighting that commonality can have positive externalities on corporate overhead.

Benefits in testing and commissioning result from learning curves during repeated tests, amortized capital expenditure, and the potential for direct reuse of regulatory compliance tests. In the transportation and aviation markets, these benefits can be significant—reuse of an aircraft type certificate can save years in time to market.

Benefits in the operation phases are analogous to the benefits in the prior four phases. Table 2.2 shows a mapping of operation benefits to previous benefits, with the type indicated as a general categorization of the benefit.

Operations raise an important question about *who* benefits from commonality. For an aircraft manufacturer, which does not operate the products it produces, the benefits of commonality in operations will accrue to the operating carrier.

**Table 2.2** Comparison of analogies to operations benefits

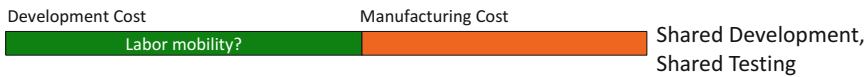
Phase	Type	Benefit	Operations Analogy
Design	Non-recurring labor	Shared development cost (intended commonality)	Reduced sustaining engineering
	Non-recurring labor	Reuse of already designed components and systems (unintended commonality)	
	Technology reuse	Reuse of proven technologies	
Manufacture	Capital	Shared tooling	Decreased fixed costs from shared facilities
	Capital	Economies of scale in manufacturing	Economies of scale in operations
	Volume	Learning curve benefits	
	Volume	Bulk purchasing	Bulk purchasing of consumables
	Volume	Reduced inventory	Decreased variable costs due to more efficient logistics and sparing
	Quality	Reduced quality expense	Slower replacement rate for spares (higher quality)
	Flexibility	Flexibility in variant volumes (for a fixed platform extent)	Flexibility in operations
Testing and commissioning	Non-recurring labor	Reduced testing and commissioning time	Decreased operator training
	Non-recurring labor	Reduced external testing/certification	Shared inspections/recurring regulatory compliance
	Capital	Shared testing equipment	

For example, airlines that operate Airbus A319, A320, and A321 aircraft can leverage the common glass cockpit instruments for shared training savings and the corresponding flexibility in pilot assignment (Brüggen and Klose 2010). While these savings will not accrue to the aircraft manufacturer directly, commonality is often used as a sales and marketing strategy. If the aircraft manufacturer can produce convincing calculations of fleet savings in operations from commonality of new aircraft with the operating carrier's existing fleet, commonality can be used as a sales advantage to boost units sold.

Having now identified the benefits of commonality, it is important to ask the question how big the benefits are. Our research (Cameron 2011) suggests that the benefits vary widely across industries, depending on the cost structure, clock-speed, and number of competitors. Well-executed commonality strategies can produce 15–50 % savings, while poorly executed platforms can *add* cost and overhead to products. To help understand which benefits are most likely to dominate, Fig. 2.2 illustrates two broad firm cost structures.



**Dominated by Development Cost**



**Dominated by Manufacturing Cost**

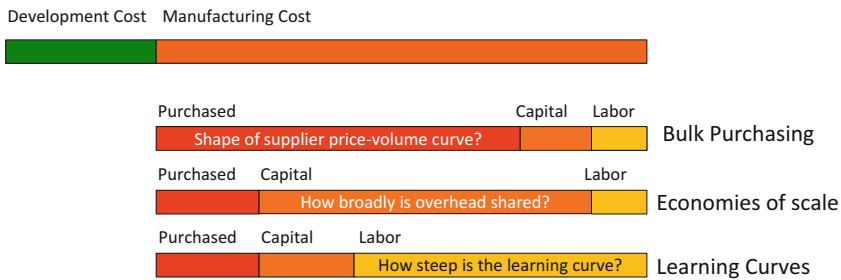


Fig. 2.2 Illustration of conceptual model of commonality benefits

**2.3.1 Industries Dominated by Development Cost**

Two criteria emerge in industries with large development cost (and typically low production volumes). The first criterion is that the saved development labor can either be productively placed elsewhere or it can be cut. It is typical to employ large-salaried workforces in several of the industries studied (e.g., Aerospace, Heavy Equipment). If the reduced head count required for later variants is not productively redeployed, the firm will not save any money. Challenges redeploying were found in organizations with high product-to-product walls and those with very dissimilar product lines.

The second criterion is that the business model does not depend on cost-plus (or similar) contracts. A number of Aerospace and Transport firms operate, or have historically operated, under design-for-fee contracts, which make it difficult to charge higher margins on later designs. This contract structure is often coupled with the practice of modifying scope or requirements (as previously discussed), which also inhibits development cost savings.

**2.3.2 Industries Dominated by Manufacturing Cost**

We propose the following three possible criteria, each of which can individually create a financially beneficial platform, although there are many possible strategies targeting individual benefits.

- Criteria 1—Significant learning curves are possible. This typically implies direct labor is a significant fraction of total lifecycle cost and also that volumes are sufficiently large to reach these learning curves. Platforms where only 1–2 %

learning curves from aggregating volumes can be achieved are unlikely to merit platform investment. Similarly, industries where configuration complexity is likely to swamp learning benefits are unlikely to retain benefits.

- **Criteria 2**—Strong bulk purchasing discounts are available. In industries that purchase a large fraction of product cost, as in Automotive, platforming will only be beneficial if there is a strong potential for a discount. If the firm cannot aggregate over sufficiently large volumes, or the suppliers have monopolies, it will be difficult to achieve a meaningful discount. In an Automotive case we conducted, several subsystems did not have sufficient visibility into their supplier’s cost structure in order to assess whether a discount could be achieved.
- **Criteria 3**—Investments in economies of scale and capital equipment will outlast the platform. Particularly in industries that are capital intensive, if the industry clock-speed dictates new manufacturing methods on short cycles, it will be challenging to invest. This is potentially the situation in semiconductor manufacturing, although Boas (2008) illustrates how, from the perspective of the manufacturer of the capital equipment (as opposed to the purchaser and user), there are sufficient projections to merit platform investment.

## 2.4 Costs of Commonality

The costs of commonality are widespread and must be carefully considered before engaging in a multiproduct strategy. Fundamentally, any commonality strategy involves significant upfront investment, in order to define the platform and create the common components. However, there are a number of costs and drawbacks that occur through the different lifecycle phases, each of which poses a risk to the successful execution of this strategy. Unrealized costs and unanticipated challenges have derailed many platforms in our experience.

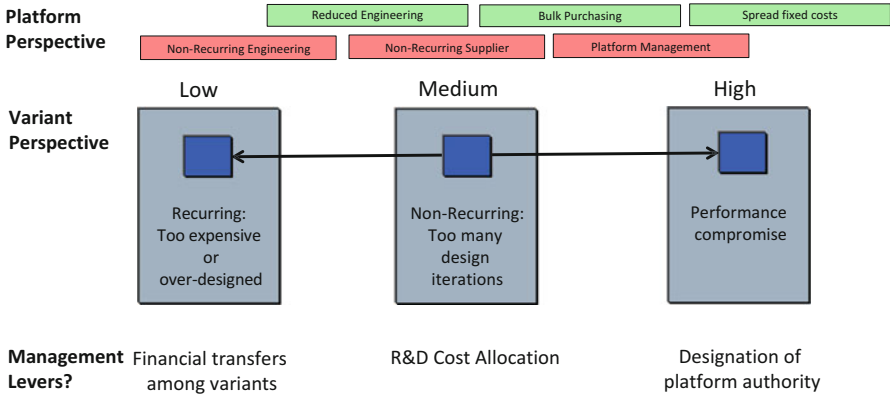
We have divided the costs and drawbacks of commonality into five categories, as with the benefits, and they are summarized in Table 2.3. This list includes both direct, quantifiable costs and broader strategic drawbacks, which are difficult to indirectly cost but represent real challenges all platforms will face. Each cost and drawback is labeled as recurring or nonrecurring with respect to additional variants. For example, the design premium is a nonrecurring cost, in that it is invested once at the beginning of the program, and can be leveraged on each variant. By contrast, the capability penalty (defined as the over-performance and cost compromises of commonality with other variants) is a recurring cost, in that it affects each variant.

Not all of these costs are expected in all commonality projects—for example, commonality may reduce the labor content in assembly, rather than increase it. This is not to say that these costs are small or easily mitigated. Most execution challenges in common programs manifest as cost problems at some point, whether it be in underestimated commonality premiums in design phases or in pro-divergence arguments based on reducing the unit cost during manufacturing.

Creating realistic projections of these costs is a competitive advantage for firms which successfully employ commonality strategies, as these projections enable the

**Table 2.3** Commonality drawbacks, costs, and risks (Note: the costs do not materialize universally; rather, the potential for costs to exist has been demonstrated, and the third column provides guidance on whether the cost recurs with each successive variant within the original platform extent or the individual cost behavior will vary by platform)

Phase	Drawbacks and costs	Recurring?	References
Strategy	Constraining future investment to platform extent	NR	Henderson and Clark (1990), Halman et al. (2003)
	Development plan risk from shared components	R	Henderson and Clark (1990)
	Brand risk from lack of differentiation	R	Kim and Chhajed (2000), Jans et al. (2008)
	Risk of cannibalization	R	Sanderson and Uzumeri (1995), Kim and Chhajed (2000)
	Risk of monopoly by common system provider	R	Swift (1995), Burke et al. (2007)
Design	Investigating technical and economic feasibility	NR	Ulrich and Eppinger (2004)
	Design premium for satisfying multiple needs	NR	Halman et al. (2003), Ulrich and Eppinger (2004)
	Costs of integration	R	Erixon and Ostgren (1993), Du et al. (2001)
	Commonality management overhead	R	Muffatto (1999), Sundgren (1999)
Manufacture	Increased cost of common items due to capability penalty (materials cost and labor cost)	R	Krishnan and Gupta (2001), Nobelius and Sundgren (2002)
	Increased complexity of configuration management on the manufacturing line	R	Thonemann and Brandeau (2000)
	Carrying costs of production assets with higher than necessary initial capacity (offset development)	NR	Thonemann and Brandeau (2000)
	Commonality management overhead	R	Muffatto (1999), Sundgren (1999)
Testing and commissioning	Cost of creating more capable test environments	NR	Halman et al. (2003)
Operation	Risk of common part failure, affecting multiple products	R	Meyer and Lehnerd (1997), Halman et al. (2003)
	Increased complexity of operating a multi-purpose item	R	de Weck (2003)
	Carrying costs of operating assets with higher than necessary initial capacity (offset development)	NR	Meyer and Lehnerd (1997)
	Commonality management overhead	R	Muffatto (1999), Sundgren (1999)



**Fig. 2.3** Arguments raised by variants that can lead to variants suboptimizing the platform

firm trade investment against the potential return and also to plan for appropriate management resources in design, manufacturing, and testing.

Past research (Ulrich and Eppinger 2004; Halman et al. 2003; Cameron 2013) suggests that the upfront investment in platforms can be multiples of an individual product design effort. If a platform of three products costs \$200 million compared with three individual products at \$100 million each, the savings are significant (\$100 million), but the initial investment is still twice the size of a typical development program. We define this initial investment as the commonality premium—the ratio of platform development cost to a single product development cost. Ulrich and Eppinger (2004) suggest 2×–10× as the premium. A subsystem-level study (Cameron 2011) in the context of a 3-case study of low-volume capital-intensive manufacturing firms indicates that the system premiums ranged from 12 % to 50 % for three platform in transportation, with subsystem premiums as high as 200 % (3× a single product subsystem development program).

These costs do accrue evenly to all products on a platform. For example, the upfront variant is likely to pay most of the commonality premium, unless the platform is explicitly structured to share investment (Meyer et al. 1997). Savings from amortized capital equipment are more likely to accrue to later variants. This imbalance implies that tensions will arise between variants—some variants will create investments that they will not be able to recover themselves. Therefore, in addition to the necessity of weighing the costs of platforming against the benefits, it is important to create a platform perspective on costs. Without a platform perspective, individual variants will systematically reject the compromises and additional costs inherent in a platform strategy in favor of lower-entropy, individualized design.

Figure 2.3 illustrates how some of these costs can be projected on to individual variants, which are arranged for a vertical platform strategy (economy to luxury products). The position of the product within the platform extent (the performance range spanned by the variants) determines which of the benefits it stands to gain, as well as which of the costs it may have preferred not to shoulder. For example, the low performance variant typically aims to minimize unit cost to provide the lowest

possible entry price into the market (de Weck 2006) and will therefore attempt to reject common components with heavy capability penalties or hooks for expensive options. Figure 2.3 illustrates the most common source of complaint for each variant in the platform extent.

## 2.5 Planning for Divergence

Despite significant investments and planning efforts, many platforms tend to realize less commonality than intended, a phenomena we call “divergence.” This phenomenon appears to affect platforms across industries, ranging from automotive to semiconductor capital equipment as summarized in Table 2.4. There is a large body of work on developing commonality metrics (Wacker and Treleven 1986; Siddique et al. 1998; Jiao and Tseng 2000; Thevenot and Simpson 2006), but descriptive studies tracking commonality indices over time are just beginning to emerge (Fixson 2007). A widely cited example is the Joint Strike Fighter, a military aircraft designed with three variants, which was intended to share 80–90 % parts commonality across all three variants. Through development and early production phases, commonality fell sharply to 30–40 % parts shared (Boas et al. 2012).

The magnitude of this phenomenon is not static across industries or platforms. Some platforms see minimal erosion of targets, while others face strong pressure to move towards unique designs. Our understanding of the challenges would suggest that divergence varies much more strongly in response to a firm’s management capabilities than in response to the market in which the firm operates.

Boas et al. (2012) illustrate that divergence is not necessarily an entirely negative phenomena. For example, an optimistically scoped platform would benefit by moving to more achievable commonality level, potentially seeing reductions in development budget and schedule. Likewise, beneficial divergence can occur in the face of unanticipated technological progress or when market requirements change during the design process. Ramdas and Randall (2008) find that uniquely designed components have higher component reliability, eschewing the design compromises associated with commonality.

However, there are also negative implications from divergence. Any movement to lower commonality levels implies more unique content, which will require design work, manufacturing planning, and operational constraints. In addition to the incremental work implied, divergence reduces the extent of the cost synergies on which many platforms were founded (Cameron 2011).

Divergence results from a number of imbalances that recur in most platforms. These imbalances occur in time, resources, volumes, and markets. Almost all platforms contain some degree of time offset, where one variant is designed and manufactured before others. This lead variant has a strong influence on the platform, often shouldering the design of many of the common parts. Difficulty understanding the future needs of latter variants can cause the lead to skew the common design closer to its needs, thus creating an opportunity for divergence when latter variants inherit the skewed parts. Similar imbalance in development

**Table 2.4** Research data categorization of observed divergence together with the range of offsets, reproduced from Boas et al. (2012)

	Automotive	Military aircraft	Commercial aircraft	Business jets	Printing press	Comm. satellite	Semiconductor capital equip.
Divergence	High	High	Moderate	Low	Moderate	High	Moderate
Offset as % of development time	100 % (24 months)	10 % (6 months)	25–280 %	0–125 %	75–250 %	0–170 %	0–130 %

budgets, expected production volumes, and perceived customer importance also creates opportunities for divergence.

Making strong decisions in the face of divergence is the result of understanding the differential impact on the benefits and costs of commonality. We've already established that all divergence has a near-term cost, due to implied unique design work, and a long-term cost, due to reduced synergies. However, the downstream positive revenue implications may dwarf the near- and long-term cost of divergence.

For example, consider a rail manufacturer attempting to produce a platform locomotive, spanning three national operating voltages. If one of those national markets changes voltages to double its existing specification, the rail manufacturer should weight the relevant implications on costs and benefits. Modifying the platform to include new operating voltage may significantly increase the commonality premium, as the design may need to be reworked. Additionally, it may raise the cost of manufacturing for all locomotives due to the capability penalty. By contrast, the manufacturer can consider diverging, creating a new locomotive targeted at one market, and reducing the existing platform specification to two voltages. This implies that there will be a lower bulk purchasing effect for the platform, because common components will not be spread across three national markets. This decision would create additional design work for the new locomotive, but it may also reduce the commonality premium for the platform, as fewer design constraints are levied. The rail manufacturer will need to weigh these costs and benefits against the revenue implications of the decision. They may in fact sell more locomotives in the remaining two national markets if they can pass the reduced commonality capability penalty on to the consumer in the form of a lower price.

Our research suggests that the firm's ability to weigh the options in a divergence decision represents a key competitive advantage for firms. Cameron (2011) illustrates the mechanisms by which divergence led to failed investment returns on large platforms. By contrast, firms like Volkswagen, which has pursued multiple product platforms, are continuing to achieve cost savings on the order of 30 % and lead time reductions on the order of 50 % (Pander 2012).

Having now illustrated that divergence opportunities need to be carefully weighed, we must ask the question of whether upfront planning should anticipate divergence. We have already illustrated that commonality planners should include sizeable commonality premiums in design phases, and we have identified downstream potential savings in supply chain, manufacturing, testing, and operations.

Our research suggests that estimating realistic commonality benefits is a firm competence. One Automotive firm we worked with kept detailed variant cost estimation models, which would project the design work required to produce a derivative (such as a long wheelbase model), as a function of the binned magnitude of changes and the complexity of the host platform.

Should platform managers actively slash projected savings and inflate commonality premiums to account for divergence? Should they assume an "average divergence" factor? We have not seen evidence in industry that this is an effective practice, beyond the standard practices of planning for program manager reserves

and estimating schedule risk. Rather, the approach followed by successful firms has been to keenly question commonality plans, attempting to pare the design down to retain feasible commonality levels. Recast in another light, stretch goals are an important practice, but they should be used incrementally rather than radically as applied to platforms.

## 2.6 Choosing a Platform Strategy

The choice of what to make common is at the heart of any platform strategy. Fundamentally, this choice must be grounded in technical reality. For example, it must be feasible to use the same water valve in three different radiators. However, the choice of platform strategy must be grounded in, and clearly traceable to, a set of financial advantages. This implies some degree of coordination between technical and financial decisions. For example, aggregating water valve purchasing across the firm to establish supplier orders of 10,000 rather than orders of 1,000 may enable a strong bulk purchasing discount.

In this section, we identify some of the canonical commonality strategies, and we compare them against the associated benefits. In parallel with this analysis, it is important to conduct the market research and planning to establish differentiation across the product family, but for the purpose of linearity, this is not discussed in detail here.

Table 2.5 lists a subset of the available platforming strategies, arranged from low commonality planning effort at the top to high commonality planning effort at the bottom. For alternative categorizations of commonality strategies, see Robertson and Ulrich (1998) and Park and Simpson (2005).

We can see from this list that pervasive commonality strategies tend to target development benefits but invest significantly up front in order to achieve this benefit. Lower-order strategies, which tend to be organization-wide rather than platform-wide (Labro 2004), are more likely to cite bulk purchasing and inventory charges. Separate from the question of whether commonality is technically feasible, it is important that the platform manager align the firm's commonality strategy with its cost structure. For example, if consolidating all the low-cost components from the firm's three product lines would double the effective volume purchased from the firm's steel supplier, the question remains whether the steel supplier would offer a discount at this volume. If the steel supplier can only make meaningful changes to cost structure based on  $10\times$  volume, then the investment in consolidating low-cost components is unlikely to bear out. Farrell and Simpson (2010) offer a methodological step in this direction, using activity-based costing to understand how consolidation of components impacts manufacturing economies.

In terms of challenges, diffuse low-order commonality strategies clearly face greater coordination challenges and specifically are more likely to face funding challenges. Higher order commonality strategies are more likely to face "execution" challenges, in terms of holding off unplanned customization (Wortmann, et al. 1997). These challenges will create divergence opportunities in all cases, whether they



**Table 2.5** Platform strategies arranged from low forward planning (top) to high forward planning (bottom)

Strategy	Applicability	Benefit	Challenges
Reactive reuse (Siddique and Repphun 2001)	Low planning ability	Development	High risk of optimal solutions
	Low R&D spending	Tooling	Potential for missed benefits
Low cost components (Labro 2004)	Flat component curve	Bulk purchasing	Hard to define fixed cost savings
	Low planning ability	Inventory	Assumes labor mobility across products
Building blocks (Fisher et al. 1999)	Stable architecture	Bulk Purchasing	Challenging to synchronize development
	High overhead	Inventory	Difficult to fund R&D
Non-differentiating subsystems	Stable architecture	Development	Managing stable interfaces
		Testing	Enabling differentiating features
High cost components (Boas 2008)	Steep component curve	Testing	Risk of high integration costs
	High R&D spend	Economies of scale	Degradation to reactive reuse
Backbone/common architecture (Halman et al. 2003)	Low clockspeed	Development	Risk to development savings—customization
	High R&D spend	Economies of scale	Does not imply testing savings
Commonality culture (Boas 2008)	High planning ability	Development	High coordination costs
	High R&D spend	Inventory	

Divergence data is binned from low to high, where low represents small changes, such as moving from 80 % of parts shared to 77 % of parts shared, and high represents changes on the order of decreases by 50 % (half of the intended common parts became unique parts). Not all calendar offsets (number of months) can be given due to confidentiality concerns (Boas et al. 2012)

manifest as product managers lobbying for exemption from high coordination costs shared via overhead or variants attempting to shirk high integration costs by moving to unique solutions. Astute program managers will also recognize that these challenges will be increasingly back-end loaded on platform timelines for higher order commonality strategies, while lower-order strategies will face more challenges upfront in aggregating diffuse product teams into ordered component strategies.

This representation of commonality strategies does not capture the complexity of the product architecture (Baldwin and Clark 2000)—it does not represent the modularity of the platform, the intended servicing functions, or the organizational implications. However, it does showcase the necessity of matching commonality strategy to an expectation of cost and benefit. Firms that attempt to commonalize as much as possible, without regard for expected benefits and implied costs, will find themselves incurring almost all of the commonality cost categories listed here and almost certainly swamping the expected benefits.

## References

- Alptekinoglu A, Corbett CJ (2008) Mass customization vs. mass production: variety and price competition. *Manuf Ser Oper Manage* 10:204–217
- Baker KRM, Magazine J, Nuttle HLW (1986) The effect of commonality on safety stock in a simple inventory model. *Manage Sci* 32:982–988
- Baldwin CY, Clark KB (2000) *Design rules, vol 1: the power of modularity*, 1st edn. The MIT Press
- Ben-Arieh D, Qian L (2003) Activity-based cost management for design and development stage. *Int J Prod Econ* 83:169–183
- Blecker T, Abdelkafi N (2006) Complexity and variety in mass customization systems: analysis and recommendations. *Manage Decision* 44:908–929
- Blecker T, Abdelkafi N (2007) The development of a component commonality metric for mass customization. *IEEE Trans Eng Manage* 54:70–85
- Boas R (2008) Commonality in complex product families: implications of divergence and lifecycle offsets. Ph.D. Thesis, MIT ESD
- Boas R, Cameron B, Crawley EF (2012) Divergence and lifecycle offsets in product families with Commonality. *Syst Eng*. doi:10.1002/sys.21223
- Bremner R (1999) Cutting edge platforms. *Financial Times Automotive World*, September
- Brüggen A, Klose L (2010) How fleet commonality influences low-cost airline operating performance: empirical evidence. *J Air Transp Manage* 16:299–303
- Burke GJ, Carrillo JE, Vakharia AJ (2007) Single versus multiple supplier sourcing strategies. *Eur J Oper Res* 182:95–112
- Cameron BG (2011) Costing commonality: evaluating the impact of platform divergence on internal investment returns. Thesis, Massachusetts Institute of Technology. <http://dspace.mit.edu/handle/1721.1/68511>
- Cameron BG, Crawley EF (2013) No panacea for platforms: challenges in product families. *California Management Review*, in process
- Clark KB, Fujimoto T (1991) *Product development performance: strategy, organization, and management in the world auto industry*. Harvard Business Press
- Collier DA (1981) The Measurement and operating benefits of component part commonality. *Decision Sci* 12(1):85–96
- Cook HE (1997) *Product management: value, quality, cost, price, profits, and organization*. Chapman & Hall
- Cusumano MA, Nobeoka K (1998) Thinking beyond lean: how multi-project management is transforming product development at Toyota and other companies. Free
- de Weck OL (2006) Determining product platform extent. In: Simpson TW, Siddique Z, Jiao J (eds) *Product platform and product family design: methods and applications*. Springer Verlag
- de Weck (2003) In: *Proceedings of DETC'03 2003 ASME design engineering technical conferences September 2–6, Chicago, Illinois USA*
- Desai P, Kekre S, Radhakrishnan S, Srinivasan K (2001) Product differentiation and commonality in design: balancing revenue and cost drivers. *Manage Sci* 47:37–51
- Du X, Jiao J, Tseng MM (2001) Architecture of product family: fundamentals and methodology. *Concurr Eng* 9:309–325
- Erixon G, Ostgren B (1993) Synthesis and evaluation tool for modular designs. In: *International conference on engineering design*, pp 898–905
- Farrell RS, Simpson TW (2010) Improving cost effectiveness in an existing product line using component product platforms. *Int J Prod Res* 48:3299–3317
- Fisher M, Ramdas K, Ulrich K (1999) Component sharing in the management of product variety: a study of automotive braking systems. *Manage Sci* 45:297–315
- Fixson S (2006) *A roadmap for product architecture costing*. Springer
- Fixson SK (2007) Modularity and commonality research: past developments and future opportunities. *Concurr Eng* 15(2):85–111
- Halman JIM, Hofer AP, van Vuuren W (2003) Platform-driven development of product families: linking theory with practice. *J Prod Innovat Manage* 20:149–162

- Henderson RM, Clark KB (1990) Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. *Adm Sci Q* 35:9–30
- Ho J, Li J (1997) Progressive engineering changes in multi-level product structures. *Omega* 25:585–594
- Jans R, Degraeve Z, Schepens L (2008) Analysis of an industrial component commonality problem. *Eur J Oper Res* 186(2):801–811
- Jiao J, Tseng MM (2000) Understanding product family for mass customization by developing commonality indices. *J Eng Des* 11:225–243
- Jiao J, Kumar A, Lim CM (2006) Flexibility valuation of product family architecture: a real-option approach. *Int J Adv Manuf Technol* 30:1–9
- Jiao J, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18:5–29
- Johnson MD, Kirchain R (2010) Developing and assessing commonality metrics for product families: a process-based cost-modeling approach. *IEEE Trans Eng Manage* 57:634–648
- Kim K, Chhajed D (2000) Commonality in product design: cost saving, valuation change and cannibalization. *Eur J Oper Res* 125:602–621
- Krishnan V, Gupta S (2001) Appropriateness and impact of platform-based product development. *Manage Sci* 47:52–68
- Labro E (2004) The cost effects of component commonality: a literature review through a management-accounting lens. *Manuf Ser Oper Manage* 6:358
- Larson PD, Kulchitsky JD (1998) Single sourcing and supplier certification: performance and relationship implications. *Indus Market Manage* 27:73–81
- Lehnerd AP (1987) Revitalizing the manufacture and design of mature global products. National Academy Press, Washington, DC
- MacDuffie JP, Sethuraman K, Fisher ML (1996) Product variety and manufacturing performance: evidence from the international automotive assembly plant study. *Manage Sci* 42:350–369
- Martin M, Hausman W, Ishii K (1998) Design for variety. In: Ho T-H, Tang CS (eds) *Product variety management*. International series in operations research and management science, vol 10. Springer US, pp 103–122
- Meyer MH (1997) Revitalize your product lines through continuous platform renewal. *Res Technol Manage* 40:17–28
- Meyer MH, Lehnerd AP (1997) The power of product platforms: building value and cost leadership. Free
- Meyer MH, Muggle PC (2001) Make platform innovation drive enterprise growth. *Res Technol Manage* 44:25–39
- Meyer MH, Tertzakian P, Utterback JM (1997) Metrics for managing research and development in the context of the product family. *Manage Sci* 43(1):88–111
- Muffatto M (1999) Platform strategies in international new product development. *Int J Oper Prod Manage* 19:449–460
- Nelson SA, Parkinson MB, Papalambros PY (2001) Multicriteria optimization in product platform design. *ASME J Mech Des* 123:199–204
- Nobelius D, Sundgren N (2002) Managerial issues in parts sharing among product development projects: a case study. *J Eng Technol Manage* 19:59–73
- Otto K, Hölltä-Otto K (2007) A multi-criteria assessment tool for screening preliminary product platform concepts. *J Intell Manuf* 18:59–75
- Otto K, Tang V, Seering W (2003) Establishing quantitative economic value for features and functionality of new products and new services. In: Belliveau P, Griffin A, Somermeyer S (eds) *PDMA Toolbook II*, pp 297–330. <http://hdl.handle.net/1721.1/3821>
- Pander J (2012) Neues Konstruktionssystem Bei VW: Gleich Ist Gut. *Der Spiegel*
- Park J, Simpson TW (2005) Development of a production cost estimation framework to support product family design. *Int J Prod Res* 43:731–772
- Pine BJ (1993) *Mass customization: the new frontier in business competition*. Harvard Business School Press

- Pine J, Ii P, Victor B, Boynton AC (1993) Making mass customization work. *Harv Bus Rev* 71:108–119
- Ramdas K (2003) Managing product variety: an integrative review and research directions. *Prod Oper Manage* 12:79–101
- Ramdas K, Randall T (2008) Does component sharing help or hurt reliability? An empirical study in the automotive industry. *Manage Sci* 54:922–938
- Ramdas K, Fisher M, Ulrich K (2003) Managing variety for assembled products: modeling component systems sharing. *Manuf Ser Oper Manage* 5:142–156
- Rhodes R (2010) Application and management of commonality within NASA systems. Master's Thesis, MIT, Cambridge, MA
- Robertson D, Ulrich K (1998) Planning for product platforms. *Sloan Manage Rev* 39:19–32
- Rothwell R, Gardiner P (1990) Robustness and product design families. In: Oliver M (ed) *Design management: a handbook of issues and methods*, vol 3. Basil Blackwell, Cambridge, MA, pp 279–292
- Rungtusanatham MJ, Salvador F (2008) From mass production to mass customization: hindrance factors, structural inertia, and transition hazard. *Prod Oper Manage* 17:385–396
- Sabbagh K (1996) 21st century jet: the making and marketing of the Boeing 777, vol 162. Scribner. <http://www.getcited.org/pub/103309083>
- Sanderson S, Uzumeri M (1995) Managing product families: the case of the Sony walkman. *Res Policy* 24:761–782
- Siddique Z, Repphun B (2001) Estimating cost savings when implementing a product platform approach. *Concurr Eng* 9(4):285–294
- Siddique Z, Rosen DW, Wang N (1998) On the applicability of product variety design concepts to automotive platform commonality. In: ASME design engineering technical conference, DETC1998/DTM, vol 5661
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf* 18:3–20
- Suárez FF, Cusumano MA, Fine CH (1991) Flexibility and performance: a literature critique and strategic framework. MIT Sloan School White Paper 3298-91-BPS
- Sundgren N (1999) Introducing interface management in new product family development. *J Prod Innovat Manage* 16:40–51
- Swift CO (1995) Preferences for single sourcing and supplier selection criteria. *J Bus Res* 32:105–111
- Thevenot H, Simpson TW (2006) Commonality indices for product family design: a detailed comparison. *J Eng Des* 17(2):99–119
- Thevenot HJ, Simpson TW (2007) A comprehensive metric for evaluating component commonality in a product family. *J Eng Des* 18:577–598
- Thonemann UW, Brandeau ML (2000) Optimal commonality in component design. *Oper Res* 48(1):1–19
- Triantis AJ (2000) Real options and corporate risk management. *J Appl Corp Fin* 13:64–73
- Ulrich KT, Ellison DJ (1999) Holistic customer requirements and the design-select decision. *Manage Sci* 641–658
- Ulrich KT, Eppinger SD (2000) *Product design and development*. Irwin McGraw-Hill, Boston
- Ulrich KT, Eppinger SD (2004) *Product design and development*. McGraw-Hill/Irwin, Boston
- Ulrich K, Randall T, Fisher M, Reibstein D (1998) Managing product variety. In: Ho T-H, Tang CS (eds) *Product variety management*. International series in operations research and management science, vol 10. Springer US, pp 177–205
- Utterback JM, Meyer MH (1993) The product family and the dynamics of core capability. *Sloan Manage Rev* 34:29–47
- Wacker JG, Treleven M (1986) Component part standardization: an analysis of commonality sources and indices. *J Oper Manage* 6:219–244
- Wicht AC, Crawley EF (2012) Relieving joint pain: planning government acquisition of complex common systems. *Def Acquis Ref J* 19:221–248
- Wilson S, Perumal A (2009) *Waging war on complexity costs*. McGraw-Hill
- Wortmann JC, Muntslag DR, Timmermans PJM (1997) *Customer-driven Manufacturing*. Chapman & Hall

# Chapter 3

## Multidisciplinary Domains Association in Product Family Design

Hoda ElMaraghy and Tarek AlGeddawy

**Abstract** This chapter presents an innovative new model for integrating the diversity of market segments requirements with the design of product families and platforms for achieving mass customization. It is hypothesized that the relationship between product design features, product functionalities, and customer requirements domains is analogous to species co-speciation in nature. Each “Market Species” represents the needs of a market segment in the customer domain, satisfied by a group of product functionalities that are associated with a group of product components forming the corresponding “product Species” (variant) in the physical domain. Co-speciation is studied in biology using the reconciliation of cladogram trees, which result from cladistic analysis of the studied species characteristics. Cladistics is used in this work to build products platform and modules which correspond to the common regional requirements of the market. Design Structural Matrices are used to capture the relationships between the three domains, while liaison graphs help avoid infeasible combinations of product components and infer possible components integration. This model is useful in products mass customization applications, where delayed product differentiation is a prerequisite, as it allows synchronizing the differentiation points in different domains to maximize the benefit from commonality of requirements, functions, and components.

### 3.1 Product Variety Management

Variety in product design arises due to the diverse needs of customers in different market segments. Since neither markets are homogeneous nor demands for products are stable or predictable, companies seek to increase their market share

---

H. ElMaraghy (✉) • T. AlGeddawy  
Intelligent Manufacturing (IMS) Centre, University of Windsor, ON, Canada  
e-mail: [hae@uwindsor.ca](mailto:hae@uwindsor.ca)

and profits by targeting many market segments. This leads to increasing product variety through design and production. In addition, product components variety always exists, which can be seen in both complex products such as automobiles and airplanes as well as simple products such as light bulbs. Since most products consist of modules, subassemblies, components, and parts, the effect of this variety propagates downwards across the different assembly levels of each product variant. Proliferation of product variation and the consequential variety levels across products structure affect all related manufacturing activities, specifically design and processing in manufacturing (Lee and Billington 1994; ElMaraghy 2009). In order to mitigate the negative effects of increased variety, many companies seek a measure of commonality by grouping their products into families and developing platform-based product variants to help increase their products variety and meet the diverse customer needs while achieving economy of scale (Jiao et al. 2007).

From the engineering perspective, a product family is designed to address customer requirements such that product variants can be both technologically feasible and desired by the market. Martin and Ishii (2002) defined design for variety (DFV) as a series of structured methodologies to help design teams reduce the impact of variety on the life-cycle costs of a product. They identified commonality as a prerequisite to establishing a product family architecture in addition to sharing a common platform. Commonality is part of the solution which reduces the complexity of product variants design and development in an environment characterized by frequent changes and short product life cycles. A product platform is a set of common elements (parts, components, processes, sequences, etc.) embodying the underlying core technology from which a stream of derivative products can be efficiently derived and launched (Simpson 2004). Product platforms accelerate product development, reduce product development costs, increase product reliability, increase variety, reduce managerial complexity, and enhance business strategy flexibility (Muffatto and Roveda 2000). Modular design is often associated with product platform design where products are composed of modules of structural elements with identifiable functions. Components of product modules are strongly interconnected, but they are weakly connected to components in other modules (Pandremenos et al. 2009). Modular design architecture supports the development of new products quickly using alternative modules or module instances. Common modules shared by more than one product variant reduce design time and cost (Ulrich and Tung 1991). A module is changed and replaced to differentiate the main product into different variants (Jose and Tollenaere 2005).

In this chapter, it is hypothesized that the dependency of the design features and customer domains requirements is analogous to the co-speciation process that takes place in nature between two coexisting groups of different species. Each market segment is akin to a species, the needs of which form its genetic structure. Such needs must be satisfied by product features which in turn form the genetic structure of the corresponding product variants. Co-speciation is studied in biology using the reconciliation of cladogram trees, which result from cladistic analysis of the studied species characteristic data. It is proposed to use cladistics similarly to build product components modules that correspond to the common regional requirements of

certain markets, albeit heterogeneous. Commonality indices and measures were used in literature for assessing either the overall modularity in a family of products or the commonality of a component across a family of products or a set of modules. They include the relative modularity measure (Gershenson et al. 1999) and the singular value modularity index (Höltkä-Otto and de Weck 2007). Unlike the presented cladistics model, such commonality assessments are used to redesign and improve existing products design. Commonality indices also play a fundamental role in evaluating alternate design solutions for product families. This chapter introduces a model which integrates market analysis, design, and manufacturing to design the most economical family of products to manufacture. It maximizes common components among product variants, meets customer requirements in the targeted market segments, and helps plan the assembly of the product family using a biological analogy.

### 3.2 Biological Association Analogy

The world of artifacts in manufacturing includes many constituents, e.g., market segments, product variants, and production systems, which interact and develop over time. They behave like species that adapt to changes, affect each other, and evolve over time into new species in a manner akin to biological coevolution. In nature, interspecies interactions result in co-speciation schemes where speciation events are coupled. Analogously, in the world of artifacts, it is hypothesized that the effect of a single change in one of these constituents on others can be anticipated.

Cladistics, a classification tool extensively used in Biology (Hennig 1966, republished in 1999; Kitching et al. 1998), is used to reveal the evolution hypothesis and speciation scheme of a studied group of entities. Cladistics was first introduced to propose evolution hypotheses to product design in ElMaraghy et al. (2008). The proposed technique was applied to features of automobile engine blocks extracted from historical data and used to study their evolution and plan their future design development. This was followed by introducing the hypotheses and model that govern the mechanism of products and manufacturing systems coevolution (AlGeddawy and ElMaraghy 2010). The coevolution is examined by reconciling the pair of cladograms, known as tanglegrams, representing the evolution of belt tensioners and their assembly lines and to compare their changes. Such technique is also followed in biology to study the reciprocal coevolution of species in nature (Page 2003).

In the products design domain, the biological coevolution analogy can be observed. The degree of homogeneity of customer requirements affects the choice between integral and modular product design architectures. Designers and engineers must find the right balance between the simplicity, commonality, integration, and core processes evident in dedicated non-adaptable product platforms and the higher complexity, larger investments associated with more modularity



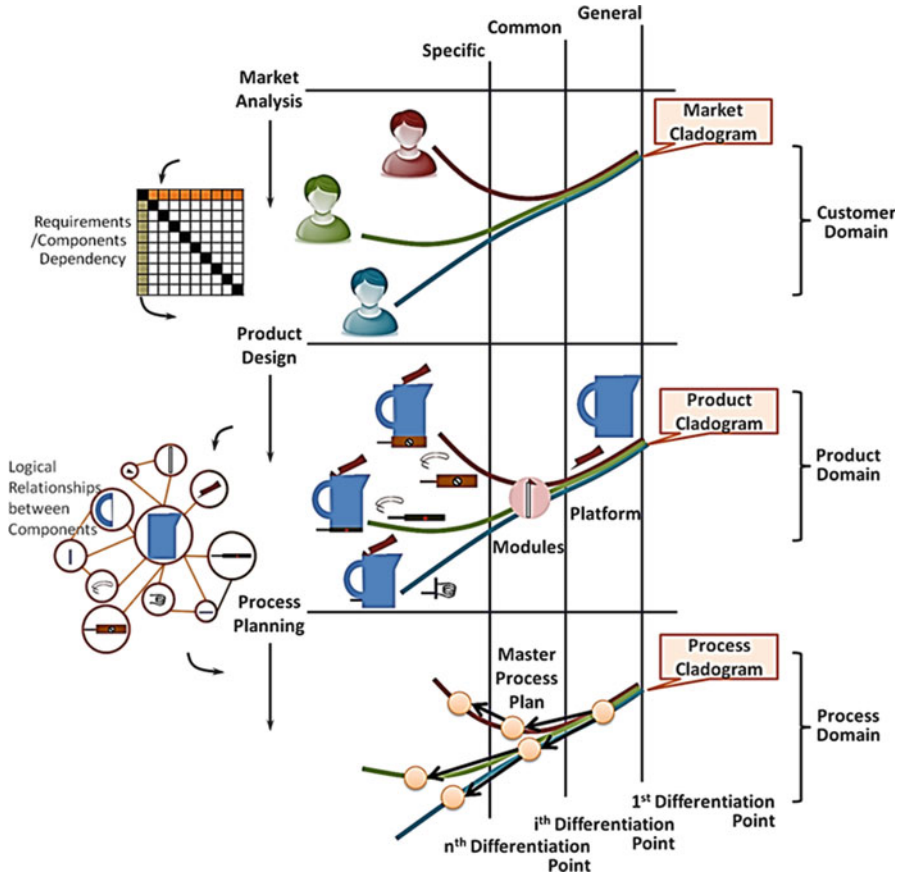


Fig. 3.1 Multidisciplinary dependency between domains in product family design

better customer satisfaction and sustainability through adaptation characterizing evolving product families and changeable manufacturing systems. The unified commonality pattern in Fig. 3.1 illustrates a recurrent foot print which relates the different constituents of manufacturing such as different market segments, different product variants, and different process plans. Such patterns are the interspecies co-speciation cladograms which support the proposed coevolution hypothesis. Identifying the market, product, and process cladograms leads to the desired unified design co-development model which links customer requirements in a market segment with product variants that satisfy them. Cladistics is usually used in Biology to study the evolution of living beings by plotting their evolution path and presenting their classification tree (Kitching et al. 1998). This powerful computational analysis results in a useful graphical clustering representation called cladogram. It shows how different entities can be grouped based on the commonality and differentiation of their characters. This ability is exploited in this research for



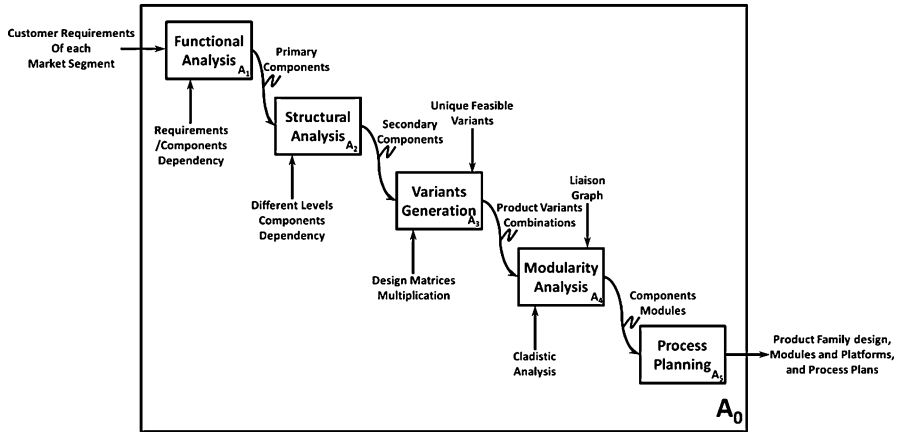


Fig. 3.2 An IDEF0 representation of the APFD model

hierarchically clustering market segments, product variants and process plans according to their commonality and showing the different differentiation points in the design and process plan of the product variants along the process flow shown in Fig. 3.1.

The associated product family design (APFD) model with market segments and process plans has been developed in (ElMaraghy and AlGeddawy 2012). It consists of five steps in Fig. 3.2: (1) *functional analysis* of customer requirements of each market segment, (2) *structural analysis* of product design to investigate all needed components, (3) *reduction analysis* of the components to be integrated into subassemblies, (4) *design analysis* for generating possible product variants designs for each market segments, and finally (5) *modularity analysis* to develop the best product variants which maximize modularity. The main objective of this new model is to develop a product variant for each market segment that (a) fulfills all its customer requirements, (b) maximizes the potential product components modules among variants, and (c) uses a common platform per product family.

### 3.2.1 Market Analysis

The diversity of market requirements is the main driver for having product design variants to address those different needs. As a preliminary step before further product design and modeling, manufacturers must identify target market segments that possess unique combinations of customer requirements to be translated into unique product variants design. There are many reasons for market segmentation and customer differentiation that can be used by manufacturers to leverage their product variant designs and meet market needs. Customer type, level of product performance, product price discrimination, different local customers taste, regional

Customer Type	Price Discrimination	Regional Standards
 Individual  Business  Government  Educational  Production	 Regular  Upgrade  Full  Quality  Luxury	

Fig. 3.3 Possible sources of market segmentation

norms and standards, etc. are but some sources of requirements diversity and product design variation in Fig. 3.3. Identification of targeted market segments is followed by recognizing the requirements of each segment. Few methods can be used to analyze customer requirements, such as quality function deployment (QFD) (Suther and Sharkey 1994), Utility functions, and design structure matrices (DSM) (Browning 2001).

The introduced APFD model uses 0–1 element DSMs to only show the existence of dependency relationships between the analyzed entities. These relationships are between two different sets of components, not between the same set of components as in the conventional DSM methods. For the design of a family of water boiling kettles in Fig. 3.4, the manufacturer identified four kettle market segments: (1) basic, (2) home, (3) office, and (4) business with different performance, appearance, and user interface. Table 3.1 shows the combinations of customer requirements suitable for each market segment. This is referred to as customer requirements matrix (R).

### 3.2.2 Functional Analysis

The first step of the APFD model is to conduct product functional analysis and construct a design structure matrix to relate the customer requirements in the identified market segments and the directly associated product components referred to in this model as “primary components.” Simple dependency relationships

**Fig. 3.4** One variant instance of the kettles product family



(existence or non-existence) are used. The DSM for market segments and kettles primary components is shown in Table 3.2. Four primary components are identified: the container, base, unit control, and control parameters input unit. Each product component has different variants/instances which may/may not exist simultaneously in the same product. In nature, this type of components classification is referred to as additive vs. nonadditive characters. Additive characters can be added simultaneously even if they are of the same type. Nonadditive characters are mutually exclusive. For kettles, variants of all components are nonadditive (i.e., they do not coexist). For example, only one On–Off switch, rheostat, or electronic board can exist as a control unit. Some customer requirements correspond to more than one entry row, which means that alternative means exist to achieve these requirements. For example, any of three alternate control units can be used to achieve unattended kettle operation.

### 3.2.3 Structural Analysis

After defining the primary product components needed to achieve the product functions corresponding to each market segment, the APFD model considers the next component levels. The product design structural analysis recognizes the relationships between primary components and other product components. This step may be repeated as many times as needed for all existing bill-of-materials (BOM) levels, until all components of all levels are connected by DSMs. Kettles secondary component level is connected to their primary components and presented in the form of a DSM in Table 3.3.

**Table 3.1** Customer requirements for the different segments of the kettles product family

Segments	Requirements										
	Targeted temperature		Human interference		Freedom of movement		Indicators		Aesthetics		
	Boil	Specific	Attended	Unattended	Fixed	Movable	Acoustic	Visual	Display	Regular	Fancy
Basic	×		×		×		×				×
Home	×			×		×		×			×
Office		×		×		×		×			×
Business		×		×	×				×		×



**Table 3.3** DSM of primary and secondary components in the kettles example

Primary		Secondary			
		Heating coil		Feedback unit	
		Side	Bottom	Condenser tube	Thermocouple
Container	Whistle				
	Window				
	Metal				
Base	Attached	×			
			×		
	Detached		×		
Control	Switch			×	
	Rheostat				
	Board				×
Input	Knob				
	Panel				

### 3.2.4 Design Feasibility and Variant Generation

The APFD model generates all possible components combinations of product variants in order to investigate all potentials for modularity. Equation 3.1 is used to generate a matrix containing all possible product variants components combinations for all designated market segments based on the previously developed DSMs.

$$P = R \cdot \pi_i^N D_i \cdot C \tag{3.1}$$

where:

*P* is the possible combinations (Variants) of product components

*R* is the customer requirements matrix

*D<sub>i</sub>* is the DSM of components of level *i* and next components level

*C* is the list of all product components primary and secondary is element-wise product operator

Some components combinations may exist under several market segments (i.e., repeated). Also, some of product components cannot mutually exist. Consequently, the resulting P matrix should be refined to remove repeated and infeasible components combinations represented by matrix P rows. In the family of kettles, the initial product variants are 4 for basic, 6 for home, 12 for office, and 24 for business market segments, for a total of 46 possible variants. However, some of these variants are repeated (identical), while others are not feasible due to the presence of nonadditive components in the same variant. After removing all redundant and infeasible variants, the number of unique feasible product variants is reduced to 8 variants in Fig. 3.5. Those variants are the possible feasible designs that can be used to establish the family of kettles which include only one variant for each market segment.

		Container			Base		Control			Input		Coil		Feedback	
		Whistle	Window	Metal	Attached	Detached	Switch	Rheostat	Board	Knob	Panel	Side	Bottom	Condenser	Thermocouple
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1.Basic	1	X			X							X			
	2	X			X								X		
2.Home	1		X			X	X						X	X	
	2		X			X			X				X		X
3.Office	1		X			X		X		X			X		
	2		X			X			X		X		X		X
4.Business	1			X	X				X		X	X			X
	2			X	X				X		X		X		X

Fig. 3.5 All possible unique feasible kettles variants

### 3.2.5 Modularity Analysis

#### 3.2.5.1 Cladistics

The APFD model searches for the best product variants under all market segments to maximize components modularity and identifies common components platforms. Parsimony analysis using cladistics (Hennig 1966, republished in 1999, Kitching et al. 1998) is used to arrange the studied product variants in a hierarchal classification tree. It minimizes the information content by identifying common components and placing them on the top branches of the tree. These common components are the possible components modules of the studied product variants.

Cladistics analysis results in a tree graph called cladogram in Fig. 3.6, which groups different entities based on their shared components (not based on similarity indices). AlGeddawy and ElMaraghy (2011) divided the process of cladogram construction into two steps: (1) cladistic tree topology construction, resulting in a binary tree where only two branches exist at any of the nodes, and (2) entities arrangement at tree terminals. This sequential two-step process simplifies the optimization of cladogram construction and makes it possible to add extra constraints or performance metrics such as precedence constraints, adjacency preferences, and assembly cost. These constraints or metrics are manufacturing related and are not needed in the classical cladograms construction used in biology.

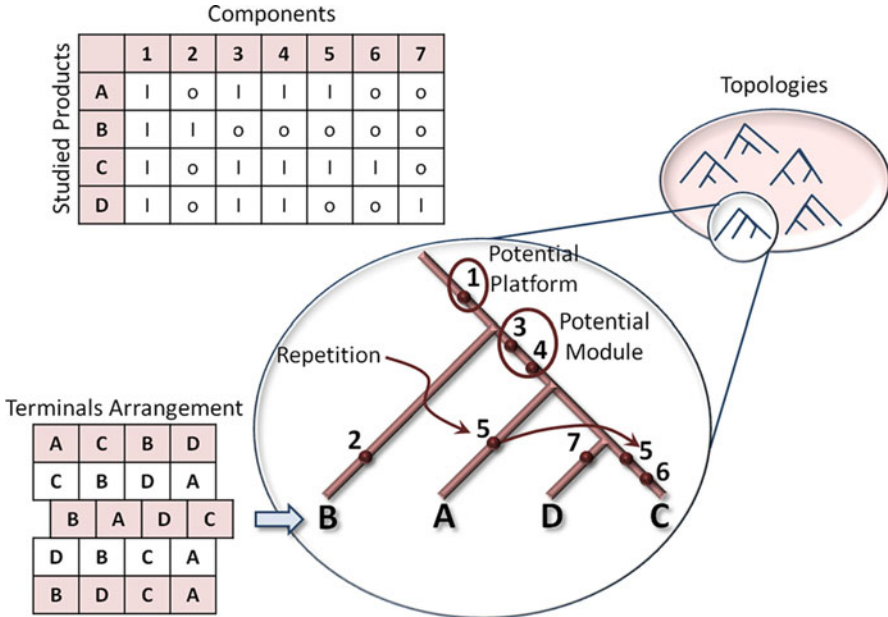


Fig. 3.6 Use of cladistics to identify product modules

The objective of the developed modified cladogram construction process is to obtain the most parsimonious data representation with minimum information content. This is referred to as the cladogram length, which is the number of characters appearing on the branches of the cladistic tree.

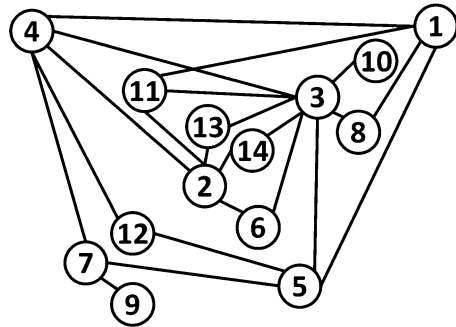
### 3.2.5.2 Identifying Product Modules and Platforms

The cladistic analysis in the developed APFD model can be illustrated using the four product variants A, B, C, and D shown in Fig. 3.6. They have a total of seven components which are not present in all entities (variants). The table in Fig. 3.6 shows the combination of components that exist in each one. The total number of all existing components in that family of products is 14 which is the upper value of the cladogram length (i.e., number of components showing on its branches), while the lower value is 7 which is the number of different components. The resulting cladogram has a length of 8 components representing a reduction of 6 ( $14 - 8 = 6$ ) from the worst case. The cladogram length is not equal to 7 (lower value) because there is data conflict, since component 5 appears twice. The objective of a cladogram construction procedure is to minimize such conflicts.

If no additional constraints or considerations exist, then the resulting cladogram presents a perfect modularity map for this family of products. The components on higher tree branches can potentially be grouped into separate modules such as



**Fig. 3.7** Liaison graph of the family of kettles



components 3 and 4, which will be a shared module in products A, D, and C. If component(s) exist at the root of the cladogram tree, which is the top most arc, then it becomes a candidate for being the base for the product family platform for further assembly, since it is common to all products, e.g., component 1 exists in products A, B, C, and D and appears at the root of the cladistic tree.

### 3.2.6 Process Planning for Product family

The unconstrained cladogram construction for a product family reveals the complete commonality map of the studied variants. However, for a group of components to form a module, connectivity between them must be present. Product components connectivity is represented in this APFD model by the adjacency constraints derived from their liaison graph. Consequently, the cladogram construction method has been modified to take into considerations such constraints. A liaison graph is a network graph representation of an assembly where nodes represent product components and arcs represent user-defined relations between assembled product components called “liaisons,” which generally include physical contact between parts (De Fazio and Whitney 1987).

Liaison graphs were mostly used for assembly process planning, optimizing assembly sequence, and generating precedence constraints for a single product variant (Sukhan 1994; Laperriere and ElMaraghy 1996; Xing et al. 2007). Dong et al. (2006) used those graphs for disassembly sequence planning. Tseng et al. (2008) used liaison graphs to improve modularity in a single product variant. In the APFD model, liaison graphs, in association with cladogram construction, are used to establish a master assembly process plan for the whole family of the studied set of products. In the kettles example, the liaison graph, comprising all existing components in those kettles and their adjacency relationships, is shown in Fig. 3.7.

Fig. 3.8 illustrates the search for product modules and the process plans using cladogram construction. The input data is divided into two types: (1) product related including components combinations of product variants, combinations of product variants, and liaison graph and (2) cladogram related including tree

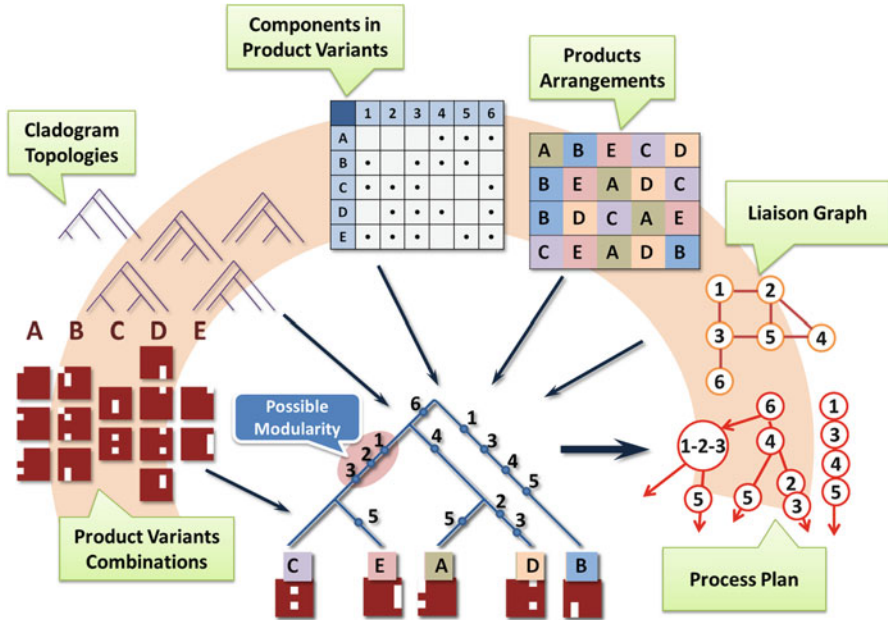


Fig. 3.8 Using cladistics for modules identification and process planning

topology and products arrangement at cladogram terminals. Components 1, 2, and 3, in the resulting cladogram, shown in Fig. 3.8, form a module according to the given liaison graph. Consequently, the cladogram length is reduced from 13 separate components to 10 components and 1 module which exist in products C and E.

Since the most common components in those products reside at the top of the cladogram and the less common ones towards the terminals of the tree, the cladogram in this form indicates the sequence of the product assembly processes that should be followed to take full advantage of the components commonality and modularity. Fig. 3.8 presents a directed tree drawn over the obtained cladogram. The directed tree in fact represents the master assembly process plan for that family of products. The resulting process plan follows the product family architecture and utilizes identified commonality as well as form postponement and delayed product differentiation strategies. The tree also serves as a schematic for the physical assembly process/system arrangement and clearly identifies the products differentiation points where work-in-process (WIP) buffers can be placed at stations represented by corresponding tree nodes. The final product variant identities are apparent at the terminals of the resulting cladogram tree/assembly process plan.

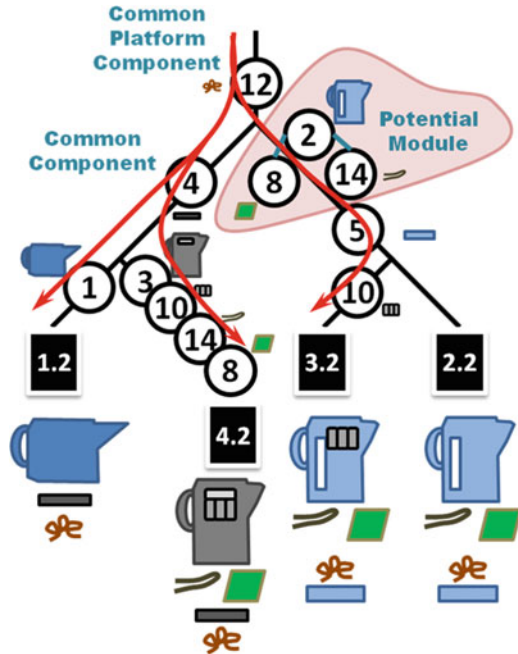
### 3.2.7 APFD Model Algorithm

The implementation of the APFD model and its algorithms for enhancing products modularity, components clustering, defining product families, and simultaneously deriving the product family process plan using the new method of cladogram construction are outlined as follows:

#### Algorithm for Modularity Promotion and Process Planning

1. Acquire:
  - T\_cladogram topology T,
  - A\_product variants terminal arrangement,
  - V\_product variants combinations,
  - C\_components of product variants {C: c=1 component exists in product variant, else c=0},
  - L\_Connectivity relationships
2.  $N_{\min} \leftarrow$  very large positive integer. Initiate optimization.  $N_{\min}$  is the minimum cladogram length
3.  $C_{ij} \leftarrow \emptyset$ . Initiate the set  $C_{ij}$  that will contain product components of level i, and position j,  $i=1, \dots, n, j=1, \dots, n+1-i$
4.  $C_{nj} \leftarrow C_{V_j A_j}$ . Determine components at nodes  $X_{nj}$  by placing associated Components  $C_{V_j A_j}$  with each product in combination V and arrangement A to position j of level n,  $j=1, \dots, n$
5.  $i = n-1$ . Consider next level
6.  $j = 1$ . Consider 1st node in the current level
7. If  $X_{ij} \neq 0$ :  $X_{ij} \square T$ . A node exists at position j in level i for topology T  
Then  $C_{ij} \leftarrow C_{i-1 j} \cap C_{i-1 j+1}$ . Search for common components at current node
8.  $E_{ij} \leftarrow |C_{ij}| - |L_{C_{ij}}|$ . Calculate information content at current node after integration and modularization.
9. If  $E_{ij}=0$  then let  $E_{ij}=1$
10.  $j \leftarrow j+1$ . Proceed to the next node in the current level
11. If  $j = i+1$   
Then Go to step 12. All nodes of current level are tested  
Otherwise repeat from step 7
12.  $i \leftarrow i - 1$ . Proceed to the next upper level
13. If  $i = 0$   
Then Go to 14. All levels are tested  
Otherwise repeat from step 6
14.  $N \leftarrow \sum_{i,j} E_{ij}$ . Calculate cladogram length
15. If  $N < N_{\min}$   
Then:
  - $N_{\min} = N$
  - Store T and A as best layout
16. Repeat from 1 until all required T and A are tested

**Fig. 3.9** Kettles design platform and master process plan representation



The previous algorithm embodies the evaluation function and the required constraints necessary to obtain the potential product modules and product family platforms as well as the directed cladogram tree and the product family master assembly process plan. Any meta-heuristic search technique may be used in the solution algorithm, since the evaluation function is separate from the search process.

In the kettles example, for 4 market segments, there exist  $4-1! = 6$  cladogram topologies and  $4! = 24$  product arrangements at the cladogram terminals, in addition to 2 product variants for each segments, which results in  $2 \times 2 \times 2 \times 2 = 16$  variants combinations. The total number of possible solutions in the solution space is  $6 \times 24 \times 16 = 2,304$ . This is a very modest solution space size that could be fully enumerated and evaluated in a fraction of a second on a 3.3 GHz processor. The resulting directed cladogram tree indicates the product modularity map and master process plan of the family of kettles (see Fig. 3.9). It represents a complete modularity map of all members of the kettles family with the most economical selection of product variants from all possible feasible variants. It maximizes the number of common components and modules without sacrificing customer requirements pertaining to each market segment. Kettle variants 1.2, 2.2, 3.2, and 4.2 were selected to serve each of the identified market segments. Container (2), board (8), and thermocouple (14) are grouped into a single module which exists in variants 2.2 and 3.2. This grouping was allowed under the constraints expressed in the liaison graph shown in Fig. 3.7. The board (8) and thermocouple (14) are also repeated in variant 4.2, but they cannot be combined into one module, since they are

not adjacent according to the liaison graph. The coil (12) appears to exist in all kettle variants and resides at the top of the cladogram; hence, it can serve as the family platform. Since only 4 variants were selected from the possible 8 variants, some components could be abandoned, such as the switch (6), rheostat (7), knob (9), side coil (11), and condenser (13).

### 3.3 Discussions and Conclusions

The heterogeneity of market segments and their customer requirements increases the complexity of product design and challenges facing companies to satisfy those requirements and plan the efficient production of many product variants. Families, modular products, and platforms design were proposed in research and in practice to manage variety in products and variation in their components. The problem of designing a set of product variants that meets market specifications and satisfy customer requirements of the designated market segments must also consider the possibility and potential of grouping components into modules in order to decrease both the design and production complexity due to products variety and reduce the cost of their assembly. The architectural and functional relationships between product components greatly affect the design process and the feasibility of product variants configurations.

This chapter introduced the notion of co-development between market segments and products design, where common requirements are associated with the common components of product family at the high level of their architecture. The common components serve as shared modules in multiple product variants. The introduced novel APFD model was shown to satisfy customer requirements of a given market, find the best modular product family design, and take components architectural constraints into account using DSMs and liaison graphs. Components existence commonality is used to minimize components redundancy and duplicate production steps and their associated cost. The use of DSMs mapped the relationships between the different design domains of customer requirements, product functions, and different components. Liaison graphs were effectively used to modify the used cladogram construction technique, and strengthen the choice of components modules, and reduce information content and components redundancy.

The APFD model was demonstrated and applied to a case study of a family of kettles design targeting four different market segments. The model identified the best product variants which satisfy all customer requirements in each market segment while maximizing components commonality, identifying family platform and modules, and presenting a master process plan for the entire kettles family. The resulting cladogram graphically represents a modularity map showing potential modules that can be shared across many variants as well as a directed tree graph showing a master assembly process plan for the whole kettles family. The constructed cladogram also shows a number of differentiation points that are used

as a base for applying form postponement and delayed product differentiation and later for assembly system layout planning.

Novel and useful results of this research are as follows: (1) a new method for products platform design for multi-domain, multi-segment products based on a co-development hypothesis inspired by natural coevolution between species, (2) a new method for generating the “master assembly process plan” for all members of the studied product, and (3) a new technique for defining the schematic layout of the physical assembly processes/stations for the family of products with clear product variants differentiation points.

## References

- AlGeddawy T, ElMaraghy H (2010) Co-evolution hypotheses and model for manufacturing planning. *CIRP Ann Manuf Technol* 59(1):445–448
- AlGeddawy T, ElMaraghy H (2011) Design of single assembly line for the delayed differentiation of product variants. *Flex Serv Manuf J* 22(3):163–182
- Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Trans Eng Manag* 48(3):292–306
- De Fazio TL, Whitney DE (1987) Simplified generation of all mechanical assembly sequences. *IEEE J Robot Autom* 3(6):640–658
- Dong T, Zhang L, Tong R, Dong J (2006) A hierarchical approach to disassembly sequence planning for mechanical product. *Int J Adv Manuf Technol* 30(5–6):507–520
- ElMaraghy H (2009) Changing and evolving products and systems – models and enablers, chapter two. In: ElMaraghy H (ed) *Changeable and reconfigurable manufacturing systems*. Springer, London, pp 25–45
- ElMaraghy H, AlGeddawy T (2012) New dependency model and biological analogy for integrating product design for variety with market requirements. *J Eng Des* 23(10–11):719–742
- ElMaraghy H, AlGeddawy T, Azab A (2008) Modelling evolution in manufacturing: a biological analogy. *CIRP Ann Manuf Technol* 57(1):467–472
- Gershenson JK, Prasad GJ, Allamneni S (1999) Modular product design: a life-cycle view. *J Integrated Des Process Sci* 3(4):13–26
- Hennig W. (1966, republished in 1999) *Phylogenetic systematics*. Urbana: University of Illinois Press
- Höltkä-Otto K, de Weck O (2007) Degree of modularity in engineering systems and products with technical and business constraints. *Concurrent Engineering* 15(2):113–126
- Jiao J, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18(1):5–29
- Jose A, Tollenaere M (2005) Modular and platform methods for product family design: literature analysis. *J Int Manuf* 16(3):371–390
- Kitching IJ, Forey PL, Humphries CJ, Williams DM (1998) *Cladistics: the theory and practice of parsimony analysis*, 2nd edn. Oxford University Press, Oxford, The Systematics Association
- Laperriere L, ElMaraghy HA (1996) GAPP: a generative assembly process planner. *J Manuf Syst* 15(4):282–293
- Lee H, Billington C (1994) Designing products and processes for postponement. In: Dasu S, Eastman C (eds) *Management of design: engineering and management perspectives*. Kluwer Academic Publishers, Boston
- Martin M, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. *Res Eng Des* 13(4):213–235

- Muffatto M, Roveda M (2000) Developing product platforms: analysis of the development process. *Technovation* 20(11):617–630
- Page RDM (2003) *Tangled trees : phylogeny, cospeciation, and coevolution*. University of Chicago Press, Chicago
- Pandremenos J, Paralikas J, Salonitis K, Chryssolouris G (2009) Modularity concepts for the automotive industry: a critical review. *CIRP J Manuf Sci Tech* 1(3):148–152
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf: AIEDAM* 18(1):3–20
- Sukhan L (1994) Subassembly identification and evaluation for assembly planning. *IEEE Trans Syst Man Cybern* 24(3):493–503
- Suther TW, Sharkey A, (1994) Customer requirements research: providing input to quality function deployment. In: *Computing and control division colloquium on customer driven quality in product design*. IEE, London, UK
- Tseng H-E, Chang C-C, Li J-D (2008) Modular design to support green life-cycle engineering. *Expert Syst Appl* 34:2524–2537
- Ulrich K, Tung K (1991) Fundamentals of product modularity. *Issues in design manufacturing/integration*. ASME 39(1):73–79
- Xing Y, Chen G, Lai X, Jin S, Zhou J (2007) Assembly sequence planning of automobile body components based on liaison graph. *Assemb Autom* 27(2):157–164

# Chapter 4

## Modular Function Deployment: Using Module Drivers to Impart Strategies to a Product Architecture

Mark W. Lange and Andrea Imsdahl

**Abstract** Products reflect the needs of many different entities. People such as end-users and re-sellers, regulating bodies of authority, and individuals within manufacturing and engineering provide statements as “voices” that impact the physical structure of a product in different ways. A company establishes a strategy to realize the product that responds to these “voices.” There are various approaches to capturing a singular “voice of x” when realizing a product architecture. Modular function deployment differs from other architecture methods by providing a holistic approach to capturing multiple “voices” as a company strategy through the use of Module Drivers. This approach is demonstrated by actual industrial examples that explore the flexibility of Module Drivers applied in the creation of a conceptual modular product architecture.

### 4.1 Introduction

To compete in today’s global marketplace, many companies are utilizing product families to increase variety, improve customer satisfaction, shorten lead-times, and reduce costs (Simpson et al. 2006).

Product families and synonymously, product platforms and product architectures, are showing themselves to be a tactic of enabling strategic objectives. The industrial community is reporting on the successes achieved through the systematic application of product architecture that support the idea of how useful product architectures are for enabling strategic intent. There is a relationship between a product family and the product development. The product development process is the tactical vehicle to convey the business strategy or strategic objectives

---

M.W. Lange (✉) • A. Imsdahl  
Department of Applied Mechanical Engineering, Industrial Engineering  
and Management, Royal Institute of Technology, SE-151 81, Södertälje, Sweden  
e-mail: [mlange@kth.se](mailto:mlange@kth.se)



of improving customer satisfaction, shorten lead-times, and reduce costs through the application of a product family.

Devising a means to model the relationship between the product development process and strategic business objectives would make it easier to explain how industrial companies could benefit from a modular product architecture. Naturally, the model needs to be adaptable not only for different business strategies driven by a product type or industry, but also adaptable to strategies that evolve with product maturity and new market emergence.

Modular function deployment (MFD) (Ericsson and Erixon 1999) is the method Modular Management uses to illustrate the relationship between product architecture and strategic objectives with the use of Module Drivers. Our suggestion is that by using Module Drivers, strategy can be imparted on the product architecture.

In the following sections, this topic will be developed. First, a brief study of product platform strategy and its role in the development of product architectures will position the development of our approach. Then a model of how Module Drivers impart strategy is constructed and explained. Based on our extensive database of modular product platforms, the model will be demonstrated using a number of industrial product architecture results. Finally, we will discuss some insights learned from the model's application and directions of continued research.

## 4.2 Background

### 4.2.1 *Strategy and Tactics in Product Development*

There is an inherent relationship between strategy and tactics, particularly in product development. Where strategy is the idea of an objective that can be realized through the application of a plan, tactics is then the execution and assessment of the outcome of that plan. For example, strategy is found in the idea of selecting between three product development approaches to tactically develop a product platform, as described in Hölttä and Salonen (2003). However, joining the idea of a strategic objective with the tactical outcome is often inferred, even when the evidence of strategy can be found in the tactical results, like when describing how product platforms are being tactically leveraged by market strategies, from Marion and Simpson (2006).

However, let us take note of the observation that product platforms and product families are being utilized to realize several business objectives; increase variety, improve customer satisfaction, shorten lead-times, and reduce costs. The strategy of a business is not to realize a product platform; product platforms and product families are a tactical means to operating a business.

When we talk about business strategy, we are referring to how businesses are managed (Treacy and Wiersema 1995). For example, a business strategy is based on an idea of what is the company going to promise its customers and then it develops a plan to bring the fulfillment of the promise to the customers, which should be an activity in a product planning process (Bowman 2006). Treacy and

**Fig. 4.1** The value disciplines abstractly represented in a three-dimensional space



Wiersema call the first part of a business strategy the Value Proposition and the second part the Value-Driven Operating Model. These two parts are combined to form the desired way of managing a business, the Value Discipline. There are three Value Disciplines available to a business strategy: Product Leadership, Operational Excellence, and Customer Intimacy as shown in Fig. 4.1. Typically, a business will pick one of these value propositions as its main focus and then seek to reach a minimum level with the other two (Nightingale and Srinivasan 2011).

#### 4.2.2 *Module as a Tactical Vehicle*

In reviewing the literature on product platforms, product families, and product architectures, the concept of a module appears regularly as a tactical vehicle in new product development. The concept of a module has been labeled as a “system partition” (Christopher 1964), “structural element” (Baldwin and Clark 2000), “chunk” (Ulrich and Eppinger 2008) or “building block” (Vos 2001). Definitions for these concepts typically reference at least one of the following three key conceptual attributes:

- (a) What the module contains-responding to a Voice of Customer perspective that the module will contain a function bearing technical solution that is identified as benefiting the customer?
- (b) What the physical limits of the module are – representing the Voice of Engineering in its need to manufacture modules so that they properly fit together?
- (c) Why the module exists-reflecting the Voice of Business in configuring a product variant using the module?

MFD however promotes a definition of a modular that reflects all three voices by stating that a module is *a functional building block with specified interfaces, driven by company-specific reasons* (Erixon 1998).

### 4.2.3 Modular Function Deployment

Since 1996, Modular Management has provided the service of developing modular product architectures using a systematic method called MFD. At the start of every MFD project, a core project team is identified. It is essential for the core team to be cross-functional, which has been clarified by Kono and Lynn (2007) and Wheelwright and Clark (1995). Cross-functional teams are comprised of representatives that personify their area of responsibility and experience, respectively, as the Voice of Customer, Voice of Engineer, and Voice of Business. Collectively, these personas are referred to as “Voices of X”, or “VoX.” Cross-functional teams are flexible with a strong project focus. They also bring a breadth of knowledge which can yield an integrated system solution that is not always achievable in functional or departmental teams.

These “Voices of X” are the spoken and unspoken needs, expectations, preferences, and wants of the people who constitute a given entity. Voice of Customer is a key input for product definition and the setting of a product’s value proposition. This voice is typically represented by the sales and/or marketing function. Voice of Engineering collects inputs from engineering, manufacturing, aftermarket, etc. for the execution of the value-driven process to design an appropriate product for the customer. A company’s engineering, research and design, and manufacturing functions define this voice. Voice of Business are shareholders, corporate officers, or others involved in corporate governance who determine which value discipline is crucial to the success of not only the product but the business as a whole. Project manager, platform manager, and product managers represent the Voice of Business for a MFD project.

MFD organizes the product data, information, and knowledge gathered by the core team into a collection of matrices known as the product management map (PMM), shown illustrated in Fig. 4.2. Each voice is captured in a different matrix to generate the modular product architecture. Iterations are necessary at each step to manage the trade-offs between the different voices.

MFD is composed of five basic steps (Erixon 1998), illustrated in Fig. 4.3. The first step is represented in the quality function deployment (QFD) matrix that clarifies the customer requirements (aka customer value statements) by mapping them against the product properties. Product properties are measurable and controllable entities that allow specification of the product demanded by the customer. QFD captures the Voice of Customer and allows it to influence the design of the product at the proper level of abstraction.

The functional requirements of the product are established with the use a form of functional decomposition. Functional decomposition is then utilized to define the

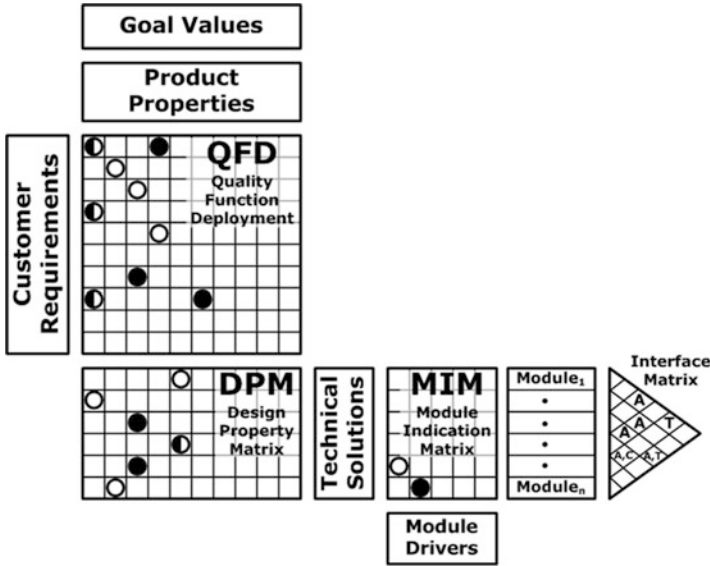


Fig. 4.2 Product management map (PMM), where the module indication matrix (MIM) and Module Drivers are the objects of this article

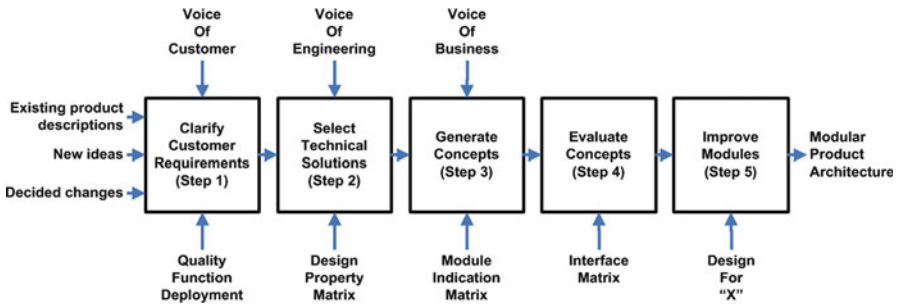


Fig. 4.3 Modular function deployment, adapted from Erixon (1998)

technical solutions. Technical solutions are the embodiment of the product properties. If necessary a Pugh process can be used to evaluate and evolve technical solutions based on evaluation criteria (i.e., product properties) generated in step 1 of MFD in addition to internal considerations such as part number count and production goals. The results of these decisions are modeled in a Design Property Matrix, first presented in Nilsson (1998), which documents the relationship between product properties and technical solutions. DPM then becomes the representation of the Voice of Engineering.

Step three highlights a unique attribute of Modular Function Deployment. Unlike other architecting approaches, MFD incorporates a company’s strategic intent into the product design. Module Drivers are the mechanism used to indicate

the strategic reason a module should be created. There are 12 Module Drivers which cover the entire life cycle of a product. A driver is applied to a technical solution in the module indication matrix (MIM) to impart the strategy the company has for a Technical Solution being the foundation of a module. Clustering the MIM and DPM, module concepts are generated therefore capturing the Voice of Business.

The module concepts are evaluated in step four by considering how the modules will be physically joined together using standardized module interfaces. Interfaces represent an agreement or contract (Baldwin and Clark 2000) between modules in a product architecture. Evaluation of the interfaces is vital to ensure flexibility of the product assortment as well as allowing for concurrent engineering. The Modular Function Deployment process considers seven basic types of interfaces. An interface can be defined as an attachment, transfer, spatial, command and control, field, environmental, and user. An interface matrix documents the interface type and facilitates the analysis of interfaces.

Finally step 5 improves on the module concept with DFX approaches, for example, Design for Manufacturing and Assembly, depending on the company value-driven operating model. Module specifications are written for each module containing market requirements, technical information, and business strategy. MFD is not a replacement for component level design improvements. Detail design of the components encapsulated in a module is still required and guided by the module specifications.

#### **4.2.4 Module Drivers**

Early during the development of Modular Function Deployment, research was conducted in industry to determine the heuristics product designers applied when creating modules by contacting a number of companies who promoted their products as modular. The resultant 12 heuristics were reported by Östgren (1994) and called “Module Drivers.” Module Drivers are found to cover the entire product life cycle from introduction to growth, maturing, and decline. Module Drivers also cover a wide spectrum of “Voices of X” as the product moves through its life cycle, as illustrated in Fig. 4.4. This coverage ensures that all stakeholders in the product have a voice as well as a way to document their particular strategic intent. Because Module Drivers are seen as generic heuristics, a project team may introduce new or modified heuristics which are company specific such as financial caps, geographical constraints, governmental regulations, etc.

The “Voice of Customer” reflects the need that a product platform embodies variance. Variation should be contained in as few areas of the product as possible to be managed effectively and minimize disruptions to the whole product when introducing new variants. Delaying variation adaptation as long as possible in the production chain decreases lead times, improves supply chain, and lowers overall costs. Two Module Drivers are available to describe the Voice of Customer variance in the product architecture:

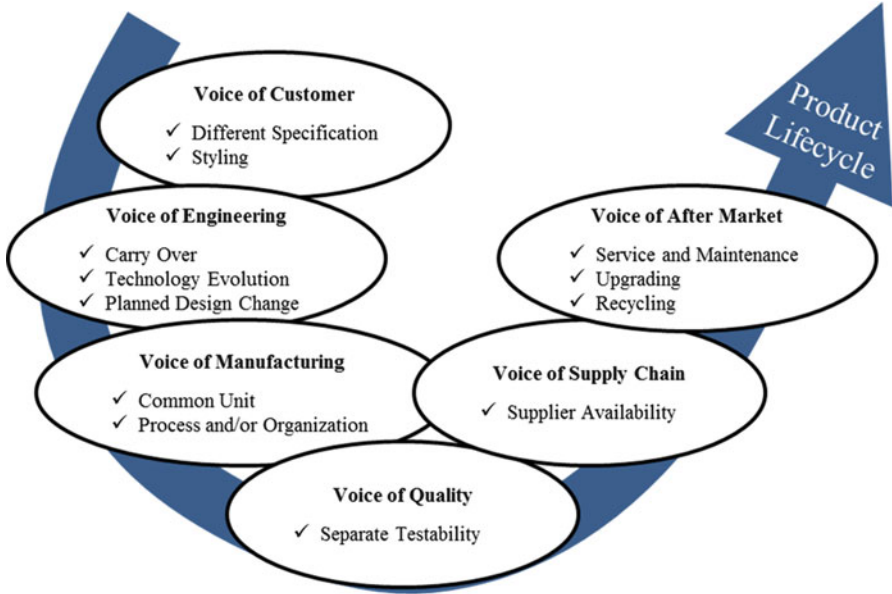


Fig. 4.4 Module Drivers positioned along a product life cycle stream

- “Different Specification” is used to impart the strategic need for technical performance variance in the product platform. Language and culture demands can change the label for the concept of the Module Driver to be referred to as “Technical Specification.”
- “Styling” imparts the strategic need for brand-driven appearance variance in the product platform.

Concerns with product planning and design are spoken through the “Voice of Engineering.” Engineering addresses the needs to manage modules of the architecture that will or will not change during the platform’s lifetime in addition to modules that will go through a technology shift based on changing customer demands. There are three Module Drivers to address engineering perspectives:

- “Carry Over” imparts strategies of technology reuse across generations of the product platform.
- “Technical Evolution” imparts the strategic development of technology driven by external forces outside the company. Language and culture demands can also re-label the concept of this driver as “Technology Push.”
- “Planned Design Changes” imparts company internal strategies to launch new products, meet changing customer requirements, or decrease product costs. An alternative name for the driver is “Planned Development.”

The “Voice of Manufacturing” strives to maintain a consistent, effective, and efficient manufacturing process. Two Module Drivers that strengthen this approach are the following:

- “Common Unit” imparts the strategy that a required function must have the same physical form in principally every product variant.
- “Process and/or Organization” imparts the strategy that there is a suitable collection of technology-driven work content for a manufacturing cell or work group to support a uniquely efficient process.

“Voice of Quality” seeks to improve the manufactured quality of a product. Increasing quality decreases the loss from warranty and product liabilities by decreasing quality feedback time. To address this concern, the following Module Drive is applied.

- “Separate Testability” or “Separate Testing” imparts strategies where functions can be tested independently of the product.

“Voice of Supply Chain” provides manufacturing with the raw material and components it needs to build the product a customer desires. At times it will be critical for an outside vendor to provide a company with standard modules or black box modules. Black box modules are modules in which a vendor takes total responsibility in terms of development, manufacture, and quality assurance. Typically, vendors of black box modules are specialists in a given technology, and a company can leverage this expertise with the use of this Module Driver:

- “Supplier Availability” imparts strategies for outsourcing “black box” technology in a module. Alternatively, this driver is re-labeled as “Strategic Supplier” or “Strategic Supplier Available.”

The addition of non-factory accessories, parts, service, or upgrades refers to the “Voice of Aftermarket.” This stage of the life cycle occurs once the product has been released to the marketplace. Sometimes, these services are offered by the company that manufactures the product and other times these services are made possible by an unrelated entity. The following three Module Drivers support this Voice of X:

- “Service and Maintenance” imparts strategies where service on a product in the field is an important customer value. The driver is also known as serviceability.
- “Upgrading” imparts strategies that will extend product life or improve product performance.
- “Recycling” imparts strategies that enable codes regarding the disposal of hazardous as well as homogenous materials.

The concept of a design strategy represented as a Module Driver for selected product life cycles is not limited to just these 12 Module Drivers. Additional drivers can be added to this generic set depending on the industry and product type. An example of the addition of a Module Driver can be found for the medical industry that develops and markets test equipment that is required to certify sub-systems with the Food and Drug Administration for products sold in the US market. A

product architecture developed for a company in this industry may well use a “Regulations Compliant” Module Driver. A similar Module Driver is also useful in the specialty vehicle industries that develop motor vehicles for over-the-road use in Europe and North America, because road certification is different in Europe contra North America.

## 4.3 Approach

### 4.3.1 *Imparting Strategy with Module Drivers*

Module Drivers are the information objects used to bridge the business strategy with the product architecture, and often several Module Drivers are applied to indicate several strategic objectives. There are combinations of Module Drivers that are compatible, which allow them to work together to enhance a module’s strategy. For example, two objectives for creating a product platform are to shorten lead times and reduce costs. The compatible pair of Common Unit and Carry Over addresses both of these goals in a single module. By carrying over the module, lead times are reduced since no time is spent on redesigning or updating the module from one product generation to the next. Common Unit modules are typically high volume modules and therefore attain a sourcing discount. Together the drivers create a coherent strategy.

A second compatible set of drivers is Service and Maintenance and Separate Testability. Service and Maintenance modules are structured in such a way when the module stops functioning, it can be removed from the product in its entirety and replaced with a fully functional new module. If the module is also Separate Testability, service can be conducted on the dysfunctional module to discover if the module can be repaired. This combination improves customer satisfaction and reduces cost by preventing extended downtimes.

Separate Testability is also compatible with Supplier Availability. Since Supplier Availability modules are black box engineered by the supplier, companies can verify the quality of these modules by testing them prior to assembly. This shortens lead times and reduces costs.

There are also conflicting drivers. These are Module Driver combinations whose strategies are mutually exclusive. Both Technical Evolution and Planned Design Change state that the content of the module will change over time, where the content of Carry Over modules will not change over time. With differing perspectives on time, Technical Evolution and Planned Design Change should not be used in conjunction with Carry Over for any given module.

Styling and Different Specification are drivers which indicate high variance modules. Common Unit on the other hand is a driver for modules where there is no variance. These drivers have conflicting strategies and should not be used in combination with each other.



The last set of conflicting drivers is Technical Evolution and Process and/or Organization. Technical Evolution modules have content that is evolving due to external sources. Process and/or Organization modules reuse a specific manufacturing process. The idea of reuse and change conflicts with each other and should not be combined in the same module.

To illustrate the application of Module Drivers, imagine a laptop. A laptop is a collection of modules each with its own strategy. Take the screen for example. Screens are offered in sizes ranging from 12 in. to 17 in. Each size offers a distinguishable performance to the customer. In contrast to the laptop, an iPad<sup>®</sup> is currently only available in one screen size. The screen, as a module, has therefore been imparted with the Common Unit driver as a business strategy.

Another example of Different Specification is the battery. A battery offered in a laptop can be available in two different performance levels: regular life and extended/heavy duty life. Depending on a consumer's use of the laptop, say someone who travels on a plane regularly, they would buy a battery with extended life so that during a flight they can work for hours without having to recharge. On the other hand, someone who only checks e-mail and surfs the web for an hour or so at night has their needs met with a regular life battery. Therefore battery supports Different Specification to meet the needs of both customers.

Battery packs found in a typical laptop can be replaced in the event the battery will no longer hold a charge. This capability would be imparted by the Service and Maintenance driver for a business strategy. However, consider the current generation of e-readers and tablets. If the battery in these devices no longer charges, the battery in the unit cannot be replaced by the consumer. This reflects a business strategy for the product architecture that does not include Service and Maintenance on the battery.

The laptop and e-readers/tablets use their case as a style design element in the architecture. Apple<sup>®</sup> has trademarked the iPad<sup>®</sup> white case with clean simple lines. It is the look that Apple<sup>®</sup> has become known for. With the introduction of the iPhone, iPads now also come in black. Similarly, laptops allow a customer to purchase a case in any color of the rainbow. This choice is imparted by applying the Styling driver.

The hinge, USB, and power button are all examples of Common Unit and Carry Over. Each of these modules has a single module variant that is used across the entire platform of laptops. Regardless if laptop has a 15" screen with regular battery life or a 17" screen with extended battery life, the hinge, USB, and power button will always be the same. While the presence of these modules is essential to the function of a laptop, none bring a tremendous amount of value to the customer. And none will be undergoing a technology shift during the lifetime of the platform. Therefore these modules also embody a Carry Over strategy.

Similar to the hinge, USB, and power button a laptops latch is also a Common Unit. It too comes in a single module variant used across all models of the product platform. Unlike the previous modules, the latch is not carried over from one product generation to the next. Instead the latch is part of the style of the laptop

case. As industrial engineers update the form of the laptop the latch will change. Therefore, the strategy of the latch is Common Unit and Styling.

Laptop microprocessors are a technology that laptop manufacturers typically do not design for themselves. The microprocessor is an example of black box engineering. The laptop manufacturers would be unable to produce the laptop without the microprocessor supplier. This is a strategic relationship between the companies and therefore a Supplier Availability module.

With a limited number of microprocessor manufacturers, there is a significant amount of competition to produce the fastest, most reliable microprocessor. Microprocessors are rapidly changing to exceed the expectation of customers. To stay in line with the latest technology laptop, companies decide if they want to enable their product architecture to handle these constant changes. If they want to offer their customers the best of what is available from the external sources, the microprocessor will also become a Technical Evolution module.

Webcams are becoming more common in laptops and tablets. For the laptop producer, the webcam will be a Technical Evolution module. However, within this webcam, there is a chip that is being developed to capture images in ever increasing resolutions. For the chip manufacture, this technical solution will be imparted with the driver Planned Design Change.

If a laptop company designs their own hard drives, it may be advantageous for them to increase aftermarket sales with an Upgrading strategy. Allowing a customer to transition from a 250 GB hard drive to a 500 GB or even a 1 TB hard drive as their storage needs change. The customer will not have to buy a new laptop to gain the extra storage, and the company will still maintain the sales revenue by selling the higher capacity hard drive.

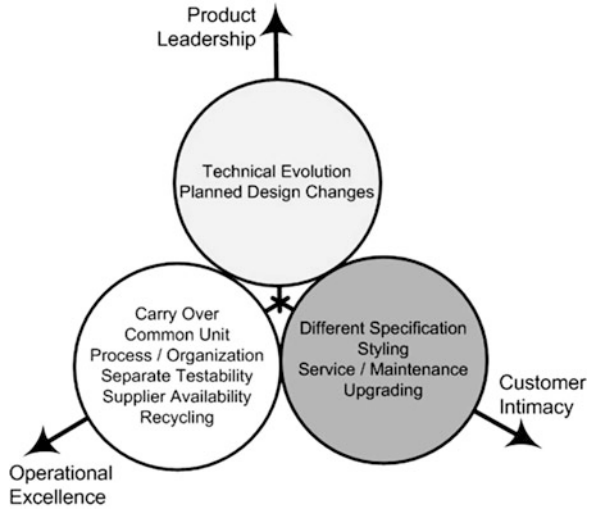
Hard drives require a clean room environment during their assembly. Any particles that are introduced may interfere and destroy the operation of the hard drive's intricate componentry. Process and/or Organization would collect the components into a module for an efficient, high quality process.

The final module is the motherboard. This module employs both Separate Testability and Recycling. Motherboards are an expensive and integral component of a laptop. Separate Testability offers quality feedback prior to the assembly of the motherboard to the case. In the event of a quality issue, the impact is contained to the motherboard only. No additional modules will need to be scrapped.

Motherboards contain materials that require special handling at the end of life of a laptop. The ability to remove the motherboard from the rest of the machine ensures that that module will be disposed of properly. This strategy is enabled by the Recycling driver.

### ***4.3.2 Aligning Module Drivers to Value Disciplines***

As information objects used to bridge business strategy with a product architecture, the application of Module Drivers impart business strategies that align with the Different Value Disciplines.



**Fig. 4.5** Value disciplines shown with the aligned Module Drivers

Product Leadership companies are creative, commercialize their products quickly, and constantly try to outdo themselves. The value proposition for a product leader is to offer their customers the best product imaginable. The Module Drivers that substantiate the value proposition are Technical Evolution and Planned Design Change. Product leaders are constantly reinventing themselves. They are not concerned if the changes are external or internal. They will lead the market regardless.

Customer Intimacy companies rely on building customer loyalty. These companies are not directly focused on a product that a market segment wants. Instead they are interested in the product a specific customer wants. The value proposition for Customer Intimacy is selling the best total solution from product to services. Different Specification, Styling, Service and Maintenance, and Upgrading all increase a company’s ability to offer the total package to a customer. Different Specification and Styling manage the customer specified variance, where Service/Maintenance and Upgrading manage the services provided once the customer has purchased the product. These drivers manage the total solution.

Operational Excellence companies are not service or product innovators. They do not build profound relationships with their customers. The value proposition for an operationally excellent organization is to provide their customers the best priced products with the least inconvenience. The Module Drivers that reinforce the proposition are Carry Over, Common Unit, Process and/or Organization, Separate Testability, Supplier Availability, and Recycling. Each of these drivers’ aides a manufacturing company in focusing on streamlining processes, reducing set-up times, and strengthening the supply chain. Reducing the cost of producing a product enables an attractive price point being offered to the customer with superior customer service.

The alignment of the Module Drivers to the Value Disciplines, as discussed, is shown in Fig. 4.5 by treating each Value Discipline as an Eulerian circle. These

alignments can be adjusted if the business strategy of company varies from the generic model. An example is found in the Module Driver for Recycling. One business strategy might place the Recycling driver along the Customer Intimacy to impart the idea that the module can be recycled by the end user as part of a “green” strategy. Another business strategy might place the Module Driver along the Operational Excellence to impart the idea of material purity in the module.

### ***4.3.3 Illustrating the Module Driver Profile***

During step 3 of a Module Function Deployment program, a project team completes the Module Indication Matrix by assigning Module Drivers to the Technical Solutions selected to be included in the architecture or platform using a scale of 9, 3, or 1. A score of 9 when applying a Module Driver represents a strong strategic interest, a 3 has a medium level of interest and a 1 has little strategic interest (but there is enough to make a difference). The scoring data is recorded in a database tool called product architecture lifecycle management (PALMA).

Each application of Modular Function Deployment that results in a module system is captured in a PALMA database that allows the results of the first three steps to be presented as a set of three linked two-dimensional matrices, also called the PMM, shown in Fig. 4.6. For this approach, we will extract the third matrix for examination, the Module Indication Matrix that illustrates the relationships between the modules and the Module Drivers.

After discussion with the company, the Module Drivers are sorted into a company specific Value Discipline that follows the general definitions, as illustrated in Fig. 4.3. These same alignments are performed on the Module Drivers in the Module Indication Matrix. Aligning the Module Drivers to the strategic axes offers a unique ability analyze the Module Driver profile for the product platform and establish the existence of alignment between the corporate strategy, i.e., value discipline and the project team strategy. The Module Driver profile is the unique combination of Module Drivers and can be found on modules as well as on a whole architecture.

In the first iteration of a Module Indication Matrix, it is difficult for the project team to commit to a single driver. Typically different voices are in conflict over the strategic intent of the module. Marketing would prefer to have a module with a high number of variants that change with each product generation, but manufacturing wants to standardize on a single solution that remains unchanged. In some cases, the project team views their corporate strategy differently than upper management. Many engineering teams believe that they are product leaders when in reality the organization has a strong focus on Operational Excellence. Multiple iterations on the Module Indication Matrix are required to reconcile these conflicts and simplify the strategy.

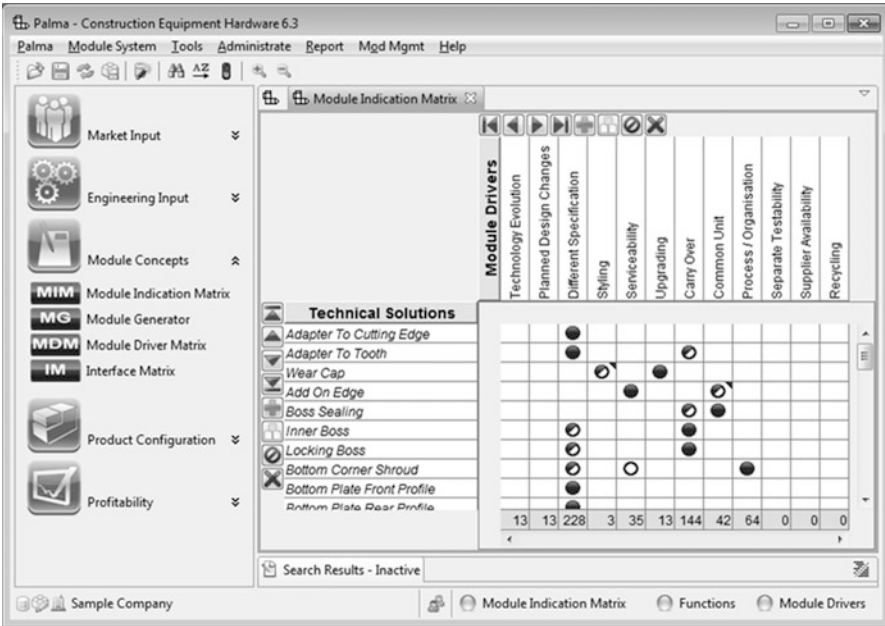


Fig. 4.6 Screenshot of the PALMA application that shows the Module Indication Matrix

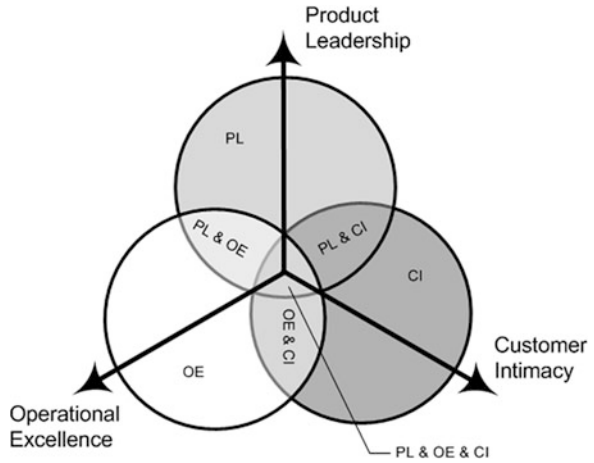
Allowing only a primary and possibly a secondary strategy reduces the complexity of a module’s design and simplifies the Module Driver profile. Think of it in terms of driving a car; would you want more than one person driving a car? A copilot comes in handy but add a backseat driver and the ride can become problematic. As with driving a car, multiple Module Drivers lead to complications in the management of the architecture. Harmonizing a module’s strategy is critical to the longevity and success of the product platform. Module Drivers capture all the voices but a decision is needed to indicate which voices are most important to the company as a whole.

Based on the Module Driver profile given for each indicated module, a count is made of the unique Module Driver profiles for each Value Discipline space and intersection. As shown in Fig. 4.7, there are only seven different spaces in which a unique Module Driver profile will appear. Depending on the application of the Module Drivers to a module it will appear in one of the seven spaces.

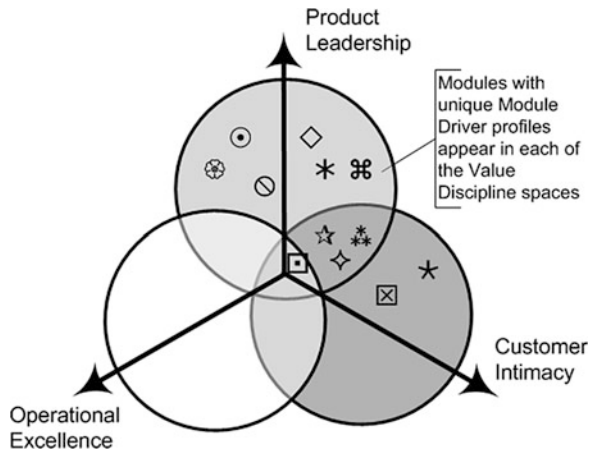
To support communication with the project team, a diagram is created that illustrates where the modules, having unique Module Driver profiles, appear in the Value Discipline spaces and intersections. An evaluation of the number of these modules and the scoring for the Module Driver profiles is performed to qualitatively understand if the stated business strategy is reflected from the distribution shown in the diagram.

An evaluation of the module system shown in Fig. 4.8 would indicate a business strategy strongly aligned with the Product Leadership Value Discipline with a weak

**Fig. 4.7** Value Discipline spaces in which Module Driver profiles will appear for a module system



**Fig. 4.8** Value Discipline spaces shown with a set of modules arranged according to their unique Module Driver profiles



alignment to the Customer Intimacy Value Discipline. This would suggest that the product platform will release a steady stream of new product variants, built on an established range of product. A product platform would have to have a very flexible and agile manufacturing process representing very little invested capital, for example, a service product.

### 4.4 Demonstrations

Since 1996 Modular Management has been building a database that contains more than 100 modular product architectures that have applied Modular Function Deployment to develop a modular product platform. The products range from microwaves to motor vehicles.

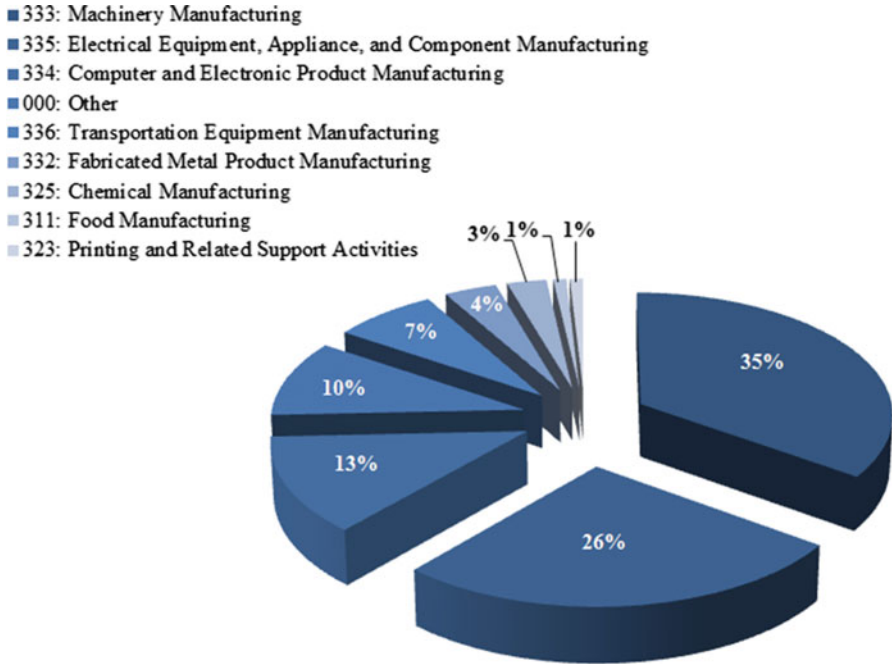


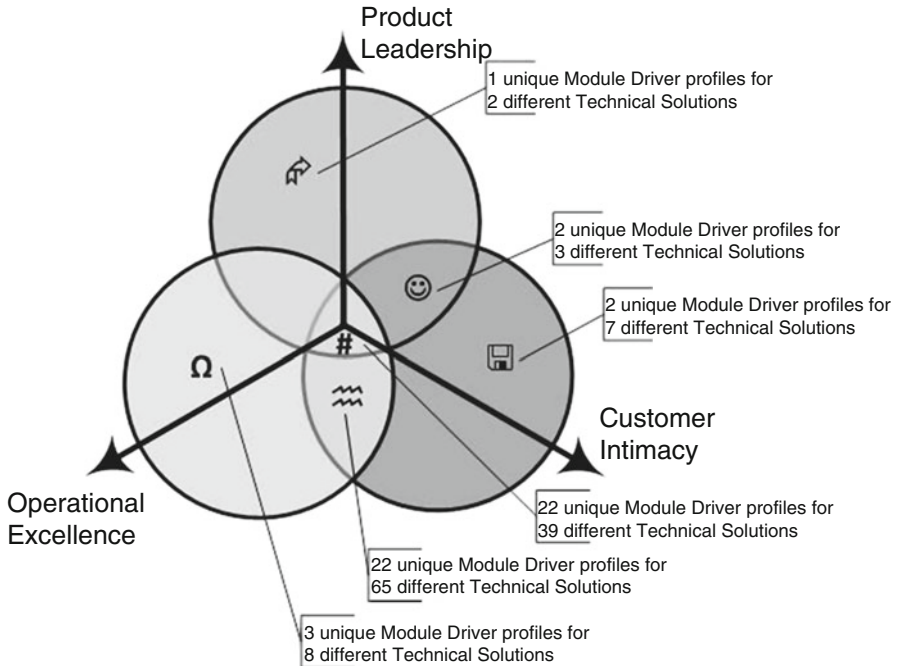
Fig. 4.9 NAICS distribution of industries that have applied Module Function Deployment

To sort this list by product types the North American Industry Classification System (NAICS) was used. The expectation was that all Module Function Deployment projects would be for companies classified within the manufacturing sector; however, only 90 % of the projects were classified as manufacturing. Looking further, the remaining 10 % of companies had a main business practice which lent itself to other NAICS codes but contained a branch dedicated to manufacturing. The distribution of products is shown in Fig. 4.9.

In the following sections, four demonstrations of Modular Function Deployment are taken from this database. Each case represents a unique situation that was illustrated using the unique application of Module Drivers to impart a business strategy to the product architecture.

#### 4.4.1 *Riding-Machine Platform*

The company in this demonstration has for over 50 years had a major focus on the Customer Intimacy Value Discipline with a minor interest in Operational Excellence. The demonstration of Modular Function Deployment for this company represents a whole vehicle platform execution; over 200 different Technical Solutions were addressed in the integration of 30 different product lines into a



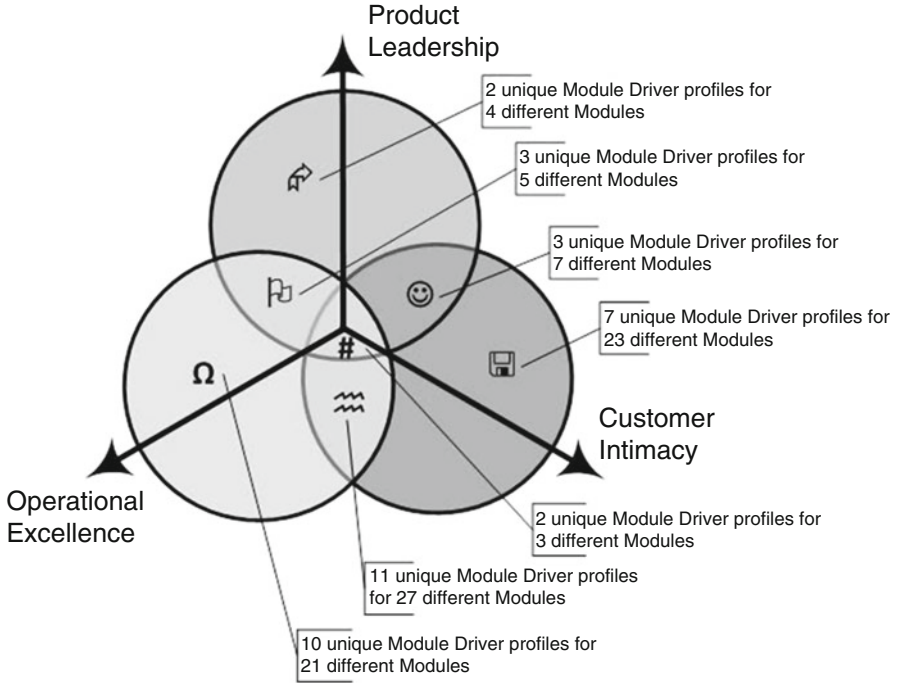
**Fig. 4.10** The Module Indication Matrix showing the initial scoring of Module Drivers for the riding-machine platform Technical Solutions

single product platform. The 30 different product lines were manufactured on no less than five different production lines in one factory and a minimum of two production lines in separate satellite factory. With the appearance of each new generation of product, a new production line was established to support that product. Furthermore, as each new generation of product was introduced, the previous generation of product was retained, including any related service and support systems. Many of the product variants were further customized after they left their respective production line.

There were two strategic objectives in applying modularity to a new product platform. The first was a technology strategy to reduce engineering effort within individual product lines by eliminating duplicate and competitive Technical Solutions that were used to fulfill the same Customer Value. The outcome expected was not just a reduction in engineering effort but also an increase in product configurability of product variants. The second was a manufacturing strategy to reduce the number of different product lines by establishing a module system that embodied more Customer Value driving variance.

Execution of step 3 of Modular Function Deployment was performed in two iterations with the first focused on imparting the project team member's current perception of the business strategy to all the Technical Solutions considered for the product platform, the data of which is shown in Fig. 4.12 and then illustrated for group discussion and evaluation in Fig. 4.10.





**Fig. 4.11** Final distribution of unique Module Driver profiles across the Value Disciplines for the technical solutions of a riding-machine platform

The second iteration was a review of the results of the first phase, with the added procedural requirement of reducing the number of applied Module Drivers to a maximum of two per Technical Solution and then integrating functionally related Technical Solutions into strategic modules, the data of which is shown in Fig. 4.13 and then illustrated for group discussion and evaluation in Fig. 4.11. This requirement was driven by the desired alignment of the product architecture to only two Value Disciplines; primary Customer Intimacy and secondary Operational Excellence.

From the results of the first phase, of the 125 different Technical Solutions considered for the product platform, over 100 were identified as belonging to two or more value disciplines, in particular the intersection of Customer Intimacy and Operational Excellence and secondarily with 39 of these Technical Solutions acquiring a Module Driver profile that spanned over all three value disciplines.

This effect of applying the Module Drivers is quite common and reflects the fact that the team members have not negotiated the tactical means of imparting the desired business strategy in a product platform. A resolution is reached by decomposing the Technical Solution into those parts that correspond best to a single value discipline or by aggregating the Technical Solution into a broader function to emphasize the importance of a single value discipline.

**Fig. 4.12** The Module Indication Matrix showing the initial scoring of Module Drivers for the riding-machine platform modules

Module Driver Combination		Product Leadership		Customer Intimacy				Operational Excellence						
		Technology Evolution	Planned Design Changes	Different Specification	Styling	Service / Maintenance	Upgrading	Affiliate Ready	Carry Over	Common Unit	Process / Organization	Separate Testability	Supplier Availability	Recycling
⊕	1													
▽	1													
⊗	2													
×	1													
%	5													
s	1													
P	1													
P	1													
⊕	1													
⊗	1													
×	1													
+	5													
⊕	1													
⊗	1													
*	1													
∨	1													
∧	1													
□	1													
•	2													
•	2													
+	4													
+	1													
⊗	3													
?	2													
▽	1													
○	6													
○	1													
†	1													
⊕	1													
⊗	7													
∇	4													
∇	4													
∇	28													
∇	3													
∇	1													
•	2													
•	2													
⊕	1													
⊕	1													
⊕	3													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	1													
⊕	6													
⊕	2													
⊕	1													
⊕	1													
⊕	2													
⊕	2													
52	124	103	223	967	15	127	114	15	693	60	235	254	30	9

**Fig. 4.13** Final distribution of unique Module Driver profiles across the Value Disciplines for the modules of a riding-machine platform

Module Driver Combination		Product Leadership	Customer Intimacy					Operational Excellence						
Module Symbol	Number of Modules	Technology Evolution	Planned Design Changes	Different Specification	Styling	Service / Maintenance	Upgrading	Affiliate Ready	Carry Over	Common Unit	Process / Organization	Separate Testability	Supplier Availability	Recycling
⊕	2													
▽	1	■	■			■				■	■	■		
⊗	2	■	■							■				
×	2	■	■											
%	1	■	■											
⋄	1								■					
⊕	1								■	■	■	■		
⊕	1								■	■	■	■	■	
⊕	1								■	■	■	■	■	
⊕	1								■	■	■	■	■	
⊕	1								■	■	■	■	■	
⊕	1								■	■	■	■	■	
⊕	2								■	■	■	■	■	
⊕	10								■	■	■	■	■	
#	1								■	■	■	■	■	
⊕	2								■	■	■	■	■	
⊕	1								■	■	■	■	■	
⊕	6								■	■	■	■	■	
⊕	2								■	■	■	■	■	
⊕	2								■	■	■	■	■	
⊕	1								■	■	■	■	■	
x	1								■	■	■	■	■	
⊕	2								■	■	■	■	■	
⊕	3								■	■	■	■	■	
*	4								■	■	■	■	■	
<	1								■	■	■	■	■	
⊕	1								■	■	■	■	■	
*	2								■	■	■	■	■	
⊕	3								■	■	■	■	■	
⊕	4								■	■	■	■	■	
+	5								■	■	■	■	■	
&	1								■	■	■	■	■	
@	5								■	■	■	■	■	
?	4								■	■	■	■	■	
⊕	1								■	■	■	■	■	
⊕	3								■	■	■	■	■	
▽	3	■	■	■	■	■	■	■	■	■	■	■	■	■
⊕	2	■	■	■	■	■	■	■	■	■	■	■	■	■
?	2	■	■	■	■	■	■	■	■	■	■	■	■	■
?	3	■	■	■	■	■	■	■	■	■	■	■	■	■
⊕	1	■	■	■	■	■	■	■	■	■	■	■	■	■
38	90	49	85	402	0	69	20	113	189	157	181	49	22	0

The platform resulted in 90 modules with roughly equal numbers on the OE (21), OE + CI (27) and CI (23) value disciplines, which accounts for about 80 % of the module system of the product platform. This was a dramatic improvement in understanding the needs of realizing the business strategy; product variance was desired, but there were still issues in the manufacturing of technology that need to be resolved to improve the performance of the product platform.

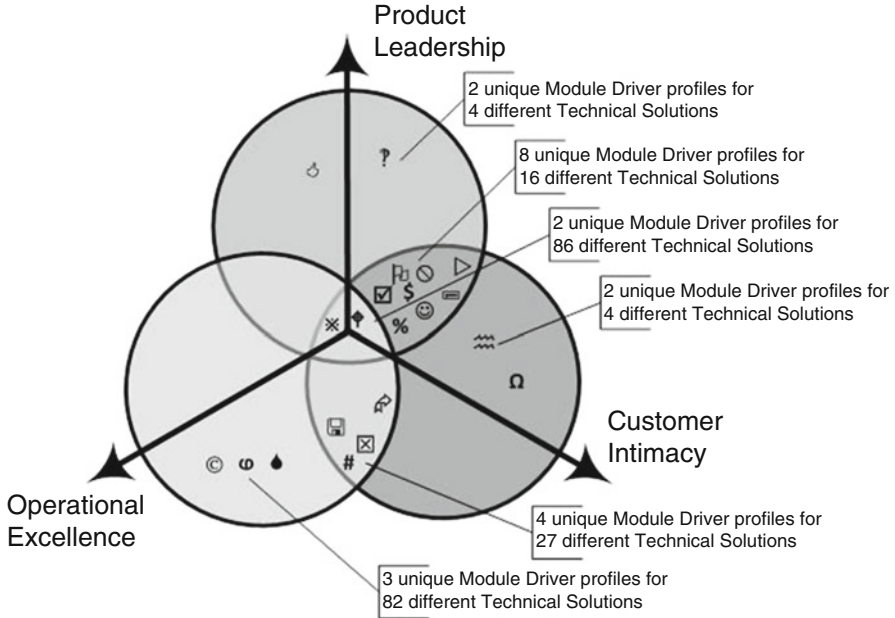


Fig. 4.14 Initial Module Driver profiles for building air conditioning system Technical Solutions

### 4.4.2 Building Air Conditioning System

The products in the building air conditioning system (BACS) market compose a landscape that has changed little over the last 75 years. Technology in the market has remained consistent forcing competitors to find other way to differentiate themselves from each other. As regions of the world are beginning to develop, opportunities for market growth are also developing particularly in the Middle East and Africa.

The project demonstrated here was looking to capture a part of this growing market. An additional challenge posed to the team was global regulations had led to the ban of a domestically used refrigerant. To seize the market opportunity, the BACS company needed to convert their existing product to the new refrigerant in the shortest amount of time possible. To achieve this goal, the platform was going to need to be focused on the value discipline of Operational Excellence.

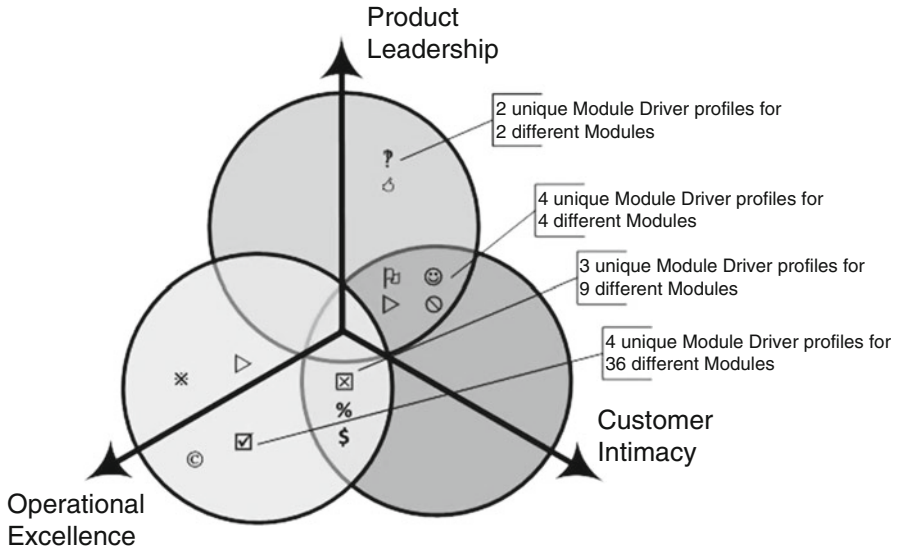
The first iteration of the platform (see Fig. 4.14 and Fig. 4.15) identified 219 modules with 21 unique Module Driver profiles. 37 % of the modules were located wholly in Operational Excellence. 12 % appeared in the overlap between Customer Intimacy and Operational Excellence. 2 % were entirely Customer Intimacy. 7 % showed up as Product Leadership and Customer Intimacy. Another 2 % was completely Product Leadership. The remaining 39 % appeared at the intersection of all three value disciplines.

Module Driver Combination		Product Leadership		Customer Intimacy			Operational Excellence						
Module Symbol	Number of Modules	Technology Evolution	Planned Design Changes	Different Specification	Styling	Service / Maintenance	Upgrading	Carry Over	Common Unit	Process / Organization	Separate Testability	Supplier Availability	Recycling
⊕	31												
●	47												
⊗	4												
+	2												
⊗	84												
⊗	9												
#	2												
⊗	5												
⊗	11												
⊗	2												
⊗	2												
⊗	1												
%	2												
S	1												
⊗	1												
⊗	1												
⊗	1												
⊗	1												
⊗	1												
⊗	1												
⊗	4												
⊗	5												
⊗	3												
⊗	1												
21	219	45	120	197	0	180	15	1149	621	0	0	0	0

Fig. 4.15 Initial Module Driver scoring for building air conditioning system Technical Solutions

To enable the program objective of launching the platform in a truncated lead time, the platform requires a strategy with an emphasis on Operational Excellence. With the initial distribution, there was a large disconnect between business strategy and team strategy. The initial distribution showed 37 % of the modules were exclusively Operational Excellence. This was not a high enough percentage to attain the quick product launch requested by management. Management realigned the team and gave them the goal of 80 % Operational Excellence.

After the realignment of the team, a second iteration of the MIM was generated. The result was 51 modules with 13 unique profiles. The new distribution had 71 % Operational Excellence, 18 % Customer Intimacy and Operational Excellence, 8 % Product Leadership and Customer Intimacy, and 4 % Product Leadership. The updated profiles were closer to the goals outlined by management (see Fig. 4.16 and Fig. 4.17).



**Fig. 4.16** Final distribution of unique Module Driver profiles across the Value Disciplines for the module system of the building air conditioning system

Module Driver Combination		Product Leadership		Customer Intimacy			Operational Excellence						
Module Symbol	Number of Modules	Technology Evolution	Planned Design Changes	Different Specification	Styling	Service / Maintenance	Upgrading	Carry Over	Common Unit	Process / Organization	Separate Testability	Supplier Availability	Recycling
⊙	30												
▽	4												
⊠	1												
×	1												
⊠	1												
%	2												
\$	6												
Ⓜ	1												
⊕	1												
▽	1												
⊙	1												
?	1												
⊙	1												
13	51	27	24	34	0	12	3	357	63	0	0	0	0

**Fig. 4.17** Final Modular Indication Matrix for the modules for the building air conditioning system platform

Module Driver Combination		Product Leadership		Customer Intimacy			Operational Excellence						
Module Symbol	Number of Modules	Technology Evolution	Planned Design Changes	Different Specification	Styling	Service / Maintenance	Upgrading	Carry Over	Common Unit	Process / Organization	Separate Testability	Supplier Availability	Recycling
◇	1												
⊙	1												
◇	4												
⊙	2												
⊕	1												
⊗	3												
▲	1												
☆	1												
⊛	1												
★	2												
⊚	1												
⊕	2												
+	1												
13	21	11	21	87	1	15	9	45	21	49	0	0	0

Fig. 4.18 The Module Indication Matrix showing the scoring of Module Drivers for the modules of the construction equipment accessory

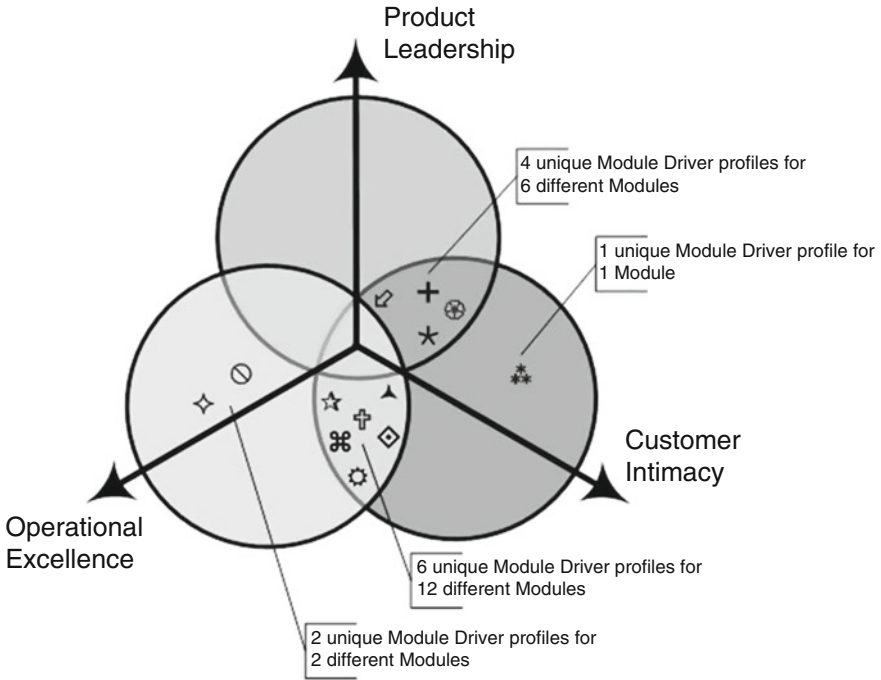
### 4.4.3 Construction Equipment Accessory

Modular Function Deployment is often applied to physical products that result in a module system assembled using mechanical fastening systems. This demonstration shows a product module system that is assembled using welding processes.

The company behind this demonstration has an established 20-year manufacturing strategy that includes outsourcing of the entire product, sub-systems, or accessories to any available regional manufacturer. The availability of “strategic suppliers” allows this company to compete with region-based third-party providers of the same accessory. Furthermore, there is a clearly stated marketing strategy of not desiring to be perceived as a Product Leader on the market, instead they intend on providing a total construction equipment solution that satisfies the needs any customer that purchases equipment.

The product platform is composed of 14 unique Module Driver profiles representing 21 different modules. The scoring is summarized in Fig. 4.18. The Module Driver profiles are distributed in the Value Disciplines shown in Fig. 4.19.

The Module Driver profile for the modules in this product platform illustrates a strong desire by the client to deliver product variants from efficient manufacturing processes. 19 of the identified modules are positioned over the Value Discipline of



**Fig. 4.19** Distribution of unique Module Driver profiles across the Value Disciplines for the module system of a construction equipment accessory

Customer Intimacy, yet 18 of these modules are located in the intersection of either the Value Discipline of Operational Excellence (12 different modules) or Product Leadership (6 different modules).

In practice, this strategy is difficult to implement. On one hand, manufacturing processes are best optimized when there is very little or no variance, and on the other hand, variance is required to fulfill customer needs. The compromise is to establish regional manufacturing lines that are optimized for the range of variants that are needed for a regional consumer.

#### 4.4.4 Cell Phone

One of the earliest applications of Modular Function Deployment was on a mobile phone, more commonly referred to as a cell phone. The client for this demonstration has a well-developed manufacturing strategy that promotes a highly automated cell-based manufacturing system intended to produce large volumes of product for delivery to a variety of different global markets. After the execution of Modular Function Deployment, 12 different strategic modules were established for the



Module Driver Combination		Product Leadership		Customer Intimacy			Operational Excellence								
Module Symbol	Number of Modules	Technology Evolution	Planned Design Changes	Different Specification	Styling	Service / Maintenance	Upgrading	Carry Over	Common Unit	Process / Organization	Separate Testability	Supplier Availability	Recycling		
△	1														
□	1														
○	1														
*	1														
⊙	1														
⊗	5														
◇	1														
⊠	1														
		8	12	21	18	12	9	9	12	9	75	0	12	0	21

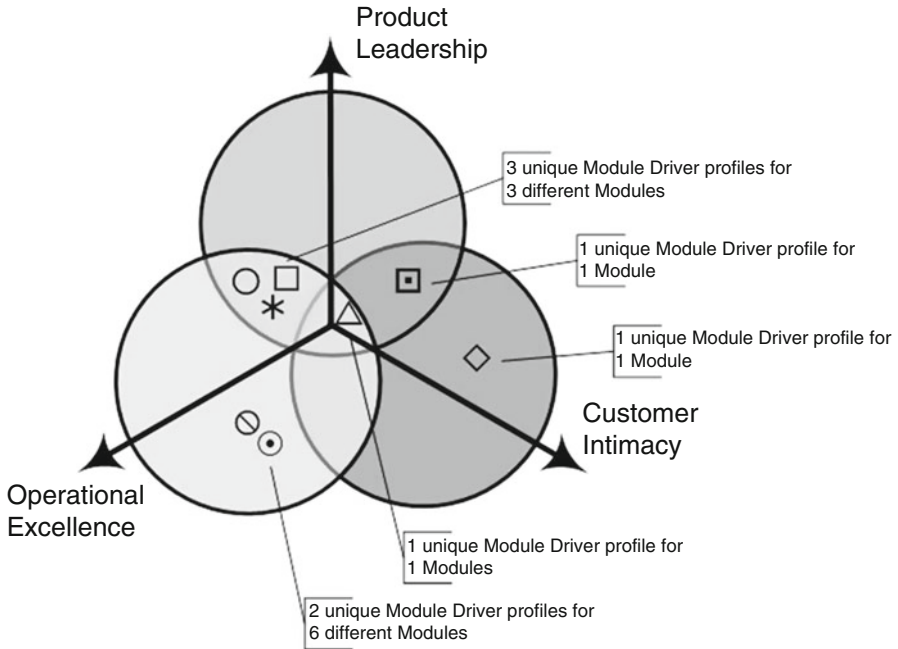
**Fig. 4.20** The Module Indication Matrix showing the scoring of Module Drivers for the modules of the cell phone

product platform, which are shown in with their scoring from the Module Indication Matrix in Fig. 4.20.

From the distribution of the unique Module Driver profiles shown in Fig 4.21, the product architecture reflects the manufacturing strategy of the client. Of the 12 modules identified for the platform, 10 of those modules are directly aligned to the Operational Excellence Value Discipline.

All of the Module Drivers, as defined, were utilized and produced eight unique combinations for the 12 different Modules in the platform. Notice that nine of these 10 modules have acquired the Common Unit Module Driver, indicating that these modules will be used in every product variant configured from the module system. Notice also that four of this set of 12 Operational Excellence Modules is contributing to the Product Leadership Value Discipline; Two modules driven by Planned Design Changes and two modules driven by Technology Evolution.

If one should emphasize the importance of responding to the strategy of providing product variance, notice that three of the 12 modules are aligned with the Customer Intimacy Value Discipline, where only one is clearly aligned with the Module Drivers for Styling plus Upgrading. This is the module intended to carry the greatest variance in the platform. However, there is also performance variance indicated using the Different Specification Module Driver for the two remaining modules.



**Fig. 4.21** Distribution of unique Module Driver profiles across the Value Disciplines for the module system of a cell phone

## 4.5 Conclusions

Industrial companies are applying product architectures to realize business objectives. In this chapter, the concept of a business strategy represented as a Value Discipline has been integrated with Modular Function Deployment's Module Drivers to show where in a product platform that a project team has imparted the business strategy. Using this approach during the execution of Modular Function Deployment, it has been possible for project teams to discern the strategic intent their decisions have made in selection of modules by reviewing the distribution over the seven different Value Discipline spaces using the Module Driver profiles.

Sometimes, there are often a great number of tough decisions that need to be made by project teams. Ensuring that the team remains focused on a primary and secondary Value Disciplines requires careful review of the application of Module Drivers and iterating on results in order to impart the desired business strategy. This makes the application of Modular Function Deployment a unique approach to the development of a modular product platform; it can be applied to devise module-based product architecture (the tactic), and it can be applied to explain the objective of the module-based architecture (the strategic).

## References

- Baldwin CY, Clark KB (2000) Design rules: vol 1, the power of modularity. MIT Press, Cambridge, MA
- Bowman D (2006) Effective product platform planning in the front end. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design; methods and applications. Springer Science + Business Media, New York, NY
- Christopher A (1964) Notes on the synthesis of form. Harvard University Press, Cambridge, MA
- Ericsson A, Erixon G (1999) Controlling design variants: modular product platforms. Society of Manufacturing Engineers, Dearborn, MI
- Erixon G (1998) Modular function deployment – a method for product modularization. Doctoral Thesis, The Royal Institute of Technology, Department of Manufacturing Systems, Stockholm, Sweden
- Hölttä KMM, Salonen MP (2003) Comparing three modularity methods. In: Proceedings of ASME design engineering technical conferences, Chicago IL, 2–6 Sept 2003
- Kono T, Lynn L (2007) Strategic New Product Development in the Global Economy. Palgrave Macmillan, New York, NY
- Marion TJ, Simpson TW (2006) Platform leveraging strategies and market segmentation. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design; methods and applications. Springer Science + Business Media, New York, NY
- Nightingale DJ, Srinivasan J (2011) Beyond the lean revolution: achieving successful and sustainable enterprise transformation. American Management Association, New York, NY
- Nilsson P (1998) The chart of modular function deployment, 4th Workshop on product structuring, Delft University of Technology, 22–23 Oct 1998
- Östgren B (1994) Modularisation of the product gives effects in the entire production. Licentiate Thesis, The Royal Institute of Technology, Department of Manufacturing Systems, Stockholm, Sweden
- Simpson TW, Siddique Z, Jiao RJ (2006) Product platform and product family design; methods and applications. Springer Science + Business Media, New York, NY
- Treacy M, Wiersema F (1995) The discipline of market leaders. Addison-Wesley Publishing, Reading, MA
- Ulrich KT, Eppinger SD (2008) Product design and development. McGraw-Hill, New York, NY
- Vos JAWM (ed) (2001) Module and system design in flexibility automated assembly. DUP Science, Netherlands, p 23, Chap. 2
- Wheelwright SC, Clark KB (1995) The Product Development Challenge: Competing through Speed, Quality, and Creativity. Harvard Business School Press, Boston, MA

# Chapter 5

## Emphasizing Reuse of Generic Assets Through Integrated Product and Production System Development Platforms

Hans Johannesson

**Abstract** Solutions from a part-based platform are inflexible to reuse in development situations as they are not allowed to be changed per definition. To use a number of such unchanged parts in new design context is problematic as related designs in the new context will be constrained. If changes are made, the initial platform intentions are violated, and economic scale benefits based on commonality may be lost. Furthermore, modifications made may result in unexpected consequences if the initial intentions and context are not properly understood. A more fruitful approach to support carryover without these drawbacks is to reuse design information containing not only final solutions but also their design rationales together with other kinds of generic assets. This is important for companies that cannot adopt a pure part-based platform approach but still want to achieve customization and economies of scale by efficient and effective reuse of other assets.

### 5.1 Background

The main driving force for platform-based development and manufacturing is the opportunity to combine customization with economies of scale. By doing this, it is possible to produce a new product variant without having to develop all of its parts, just the ones that are variant-specific. The rest can be carried over from existing products or from a common core of parts in a product family or even multiple families of different brands—the product platform. Although much has been gained with this strategy, it has its limitations. Reuse of only ready designed parts threatens to restrain further development as they per definition are not allowed to be changed, and too much commonality threatens the products' differentiation within a product offer, i.e., solutions from a part-based platform are inflexible to reuse. To carryover

---

H. Johannesson (✉)  
Department of Product and Production Development, Chalmers University of Technology,  
SE-412 96 Gothenburg, Sweden  
e-mail: [hansj@chalmers.se](mailto:hansj@chalmers.se)

a number of such parts to another design context is also problematic. If it is done, necessary modifications may lead to necessary changes of a relatively large number of other parts in the original platform, and then the economic scale benefits may be lost. Furthermore, modifications made may result in unexpected consequences if the initial intentions and context are not properly understood. A more fruitful approach that can support carryover without these drawbacks is to reuse design information described on higher levels of abstraction containing not only final solutions but also their design rationales and design histories. This kind of approach would also support reuse of other kinds of knowledge-based assets. The scope of a platform and use of its assets could then be extended beyond clean-cut configure-to-order and carryover of parts to reuse of a variety of different assets on different levels of maturity and in different development scenarios. This is important for companies that cannot adopt a pure part-based platform approach but still want to achieve customization and economies of scale benefits by efficient and effective reuse of other assets.

Researchers who have recognized the potential advantages of platform-based approaches have described and proposed different frameworks, methods, and mathematical tools to define and make use of such approaches in different industrial settings (e.g., Jose and Tollenaere 2005; Simpson 2004; Simpson et al. 2005). A well-known industrial example from Black and Decker is reported by Meyer and Lehnerd (1997) and Simpson (1998). Another industrial example described by, for example, Prencipe (1998) is from Rolls Royce. Standardization and modularization are also ways to increase reuse of resources aimed at economies of scale. A number of publications, (e.g., Baldwin and Clark 2000; Ericsson and Erixon 1999), discuss different techniques for standardization and modularization in this context.

The content of the platform and how this content is developed and described are crucial for the simultaneous optimization of both reuse and design variety as well as for data, information, and knowledge sharing. As already indicated, a platform based on parts that are already designed has severe shortcomings regarding flexibility. More abstract and easily configurable platform descriptions have therefore been proposed. van Veen (1991) proposes a generic bill of material (BOM) to provide possibilities for the description of large varieties of product types and structures. The idea is to define product data on a level of sets of product types, rather than defining individual product types. Erens (1996) applied and explored this concept further when developing product families and synthesizing product variants. In line with these ideas, Claesson (2006) has proposed a platform description approach based on autonomous configurable systems, so-called configurable components (CCs).

Reuse can benefit from product descriptions that are information-rich, i.e., descriptions that contain additional explicit information besides the resulting design. With information describing the reasoning behind the design and issues considered, referred to as the *why-information*, it is more likely that the design can be used and reused in accordance with what it was designed for. *Why-information* can bridge gaps between distances both in time and in space. Bridging distances in time means to inform about the design intent, both now and later. It also covers an

understanding how to modify the design without accidentally losing wanted behavior and, more fundamentally, how to dare to make any changes at all, instead of having to remake the complete design. Bridging distances in space deals with use in multiple contexts (i.e., use in another product than the one primarily intended), as well as concurrent activities where multiple organizations are involved simultaneously.

Collaboration between partners in a supply chain, as between an original equipment manufacturer (OEM) and its suppliers, has increased in recent years to become a natural part of the activity of any company that develops complex products. The main reason for this is that a company needs to focus on its core activities while allowing other actors to complement its shortcomings in specialized areas where suppliers can produce less expensive and better products. The products developed by suppliers have also grown over time to become complete complex systems which only need to be attached to the final product at the buyers' plant. All this puts new demands on the suppliers' own product development capabilities and involvement in the buyers' product development process.

For a supplier that is delivering customized subsystem solutions to multiple OEM companies or other higher-tier customers, a platform-based product development approach becomes as important as it is for an OEM. Both need a product platform which can be configured to deliver customized variants to several customers. An OEM delivers instantiated product variants to end customers, while a supplier, in a business-to-business relationship with an OEM, delivers customized subsystems which may require further customization by the OEM to fulfill its end customers' needs.

Suppliers' systems have become larger with greater influence on the quality and impression made of end products. Collaborative development, with information and knowledge exchange between suppliers and OEMs, has therefore become essential. Consequently, there is a growing need for new methods and tools to support the exchange of information and knowledge between collaborating partners in supply chains. Knowledge-based platforms have potential to support such exchange.

## 5.2 Platforms in Industrial Contexts

Reuse of components and interfaces has enabled large cost savings. This is the reason why product platforms have been increasingly important for many companies in recent years. Platforms enable enterprises to rapidly provide new product variants (Meyer and Lehnerd 1997; Halman et al. 2003) to respond to market changes. This facilitates more customer-oriented offers. As mentioned, platforms enable economy of scale benefits in production and efficient utilization of the resources of an enterprise. Building a platform can be seen as an evolutionary process where companies continuously have to renew their product families and, eventually, their platform to adapt to changing market needs (Meyer and Lehnerd 1997).

There are several ways to describe what a platform is. Different researchers do not define platforms and how they relate to adjacent concepts such as product

families, modules, and brands in the same way (Halman et al. 2003). Simpson (Simpson et al. 2001) define product platforms as a set of parameters, features, and/or components which remain constant for several products within a given product family. In this definition, product families are described as groups of related products that share common features, components, and subsystems, all of which can be combined to satisfy various market niches. Variants are configured by combining common components with variant-specific ones. Although this way of describing platforms is shared by many companies, it does have shortcomings. Besides the previously mentioned inflexibility problem, the inclusion of other resources such as manufacturing and organizational assets, as well as methods and IT tools in such platform definitions is difficult. A wider scope when studying platforms is exemplified by Robertson and Ulrich (1998). They describe a platform as a collection of assets, components, processes, knowledge people, and relationships that are shared by a set of products. Another way of looking at platforms is as architectures controlled by design (Gershenson et al. 2006), characterized by common structures, scaled variables, and variable structures, which can support more than one product. This view expresses the need to exchange parts or components and also to scale products to suit certain customer segments. Still another view focuses basic architectures that comprise subsystems or modules with interfaces between them (Meyer and Lehnerd 1997). Here the need for interfaces between interacting systems is emphasized.

Platforms are, according to Jiao et al. (2006), designed for either functional variety or technical variety. The first aims to satisfy diverse customer needs, while the second aims to reduce the in-house variety. Each approach requires its own strategy to address the two divergent advantages searched for in platform development, i.e., variety to enable customization or reduction of unique parts to gain economies of scale.

Many companies define a product platform as the common resources within a single product family. Other companies, such as companies in the automotive industry, are more likely to consider platforms that can carry multiple product families across different brands or across product generations (see Fig. 5.1). Here platform A may be the “small car platform” within a big automotive enterprise producing cars of different brands, whereas platform B and platform C could be the “medium-size and large-size platforms” for the companies within that enterprise (Wahl et al. 2010).

In a supply chain there is an OEM at the top producing for the consumer market. This OEM is at the same time an industrial customer buying components and subsystems from its suppliers. For an OEM producing for the consumer market, a broader platform carrying all families and brands, as shown in Fig. 5.2, would be the goal to aim for. Similarly, a supplier, delivering several variants of basically the same components and subsystems to its different customers, would benefit from the same platform approach (see Fig. 5.3).

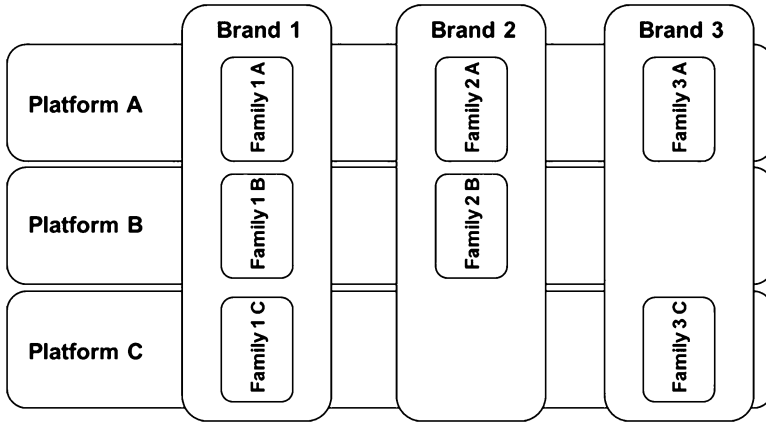


Fig. 5.1 Product platforms carrying multiple product families of different brands

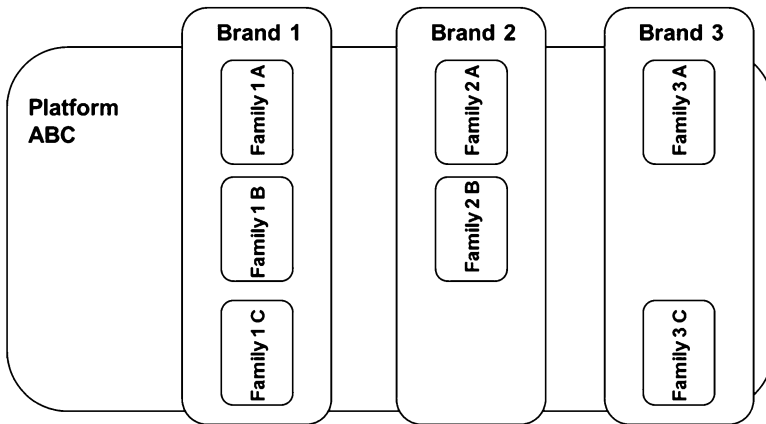


Fig. 5.2 One product platform carrying multiple product families of different brands

### 5.3 An Integrated Platform Approach

Companies that have adopted the idea of platform-based development run different kinds of product development processes depending of the level of maturity of the project outcomes. Characterization of maturity is done more or less formal in different industrial settings. In the aerospace industry, maturity (or Technology Readiness Level) is related to the NASA TRL scale (Mankins 1995), and this scale has now also been recognized within other business areas. Technology development projects, on low TRL levels (TRL 1–6), are run to learn more about and prepare new technologies for application in regular product development projects. It is a way to reduce risk in regular product development projects where



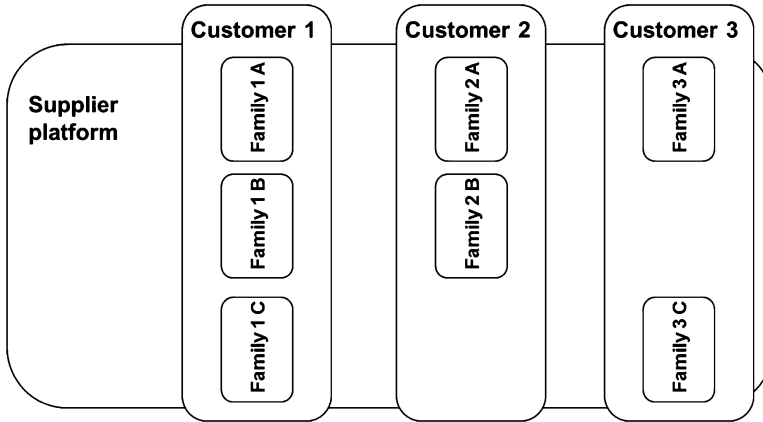


Fig. 5.3 One supplier platform carrying multiple product families for different customers

significantly bigger resources are allocated than in technology development projects. The start maturity level in product development projects can differ from *acceptable* (TRL 6 in aerospace) in product industrialization projects, to *high* (TRL 7–8 in aerospace) in engineering-change/engineer-to-order projects, and finally to more or less *complete* (TRL 8–9 in aerospace) in pure variant configuration projects. In all product development projects where the outcome is expected to be ready for production, the expected maturity of the outcome is high (TRL 8–9). In the aerospace industry TRL 9 indicates that experience has been gathered concerning operational use of the technology in an application.

In companies practicing product variant configuration, an important part of product development projects is to prepare the developed products for configuration. This means to describe the design result in such a way that it can be handled by a configuration system that can configure a master model with *bandwidth* to different variant instances within the allowed *bandwidth*. The configuration process itself, when giving variant specifying parameter values as input to the configurator system and automatically generate a variant, can be seen as a *configure-to-order process* that delivers a description of a highly mature (TRL 8–9) product variant to be produced and delivered to a specific customer.

All three kinds of above indicated development processes, technology development, product development, and product variant configuration, have outcomes that of course describe the developed technologies, products, or product variants but, following the idea of Lean product development (Kennedy et al. 2008), another important outcome is the *learning* that has taken place during the processes. Both kinds of outcomes add important knowledge value that should be properly taken care of, managed, maintained, and made available for effective and efficient reuse. This is the background for the *Development Knowledge Platform* approach proposed by the Systems Engineering & PLM group within Wingquist Laboratory at Chalmers University of Technology (WQL Chalmers) shown in Fig. 5.4.

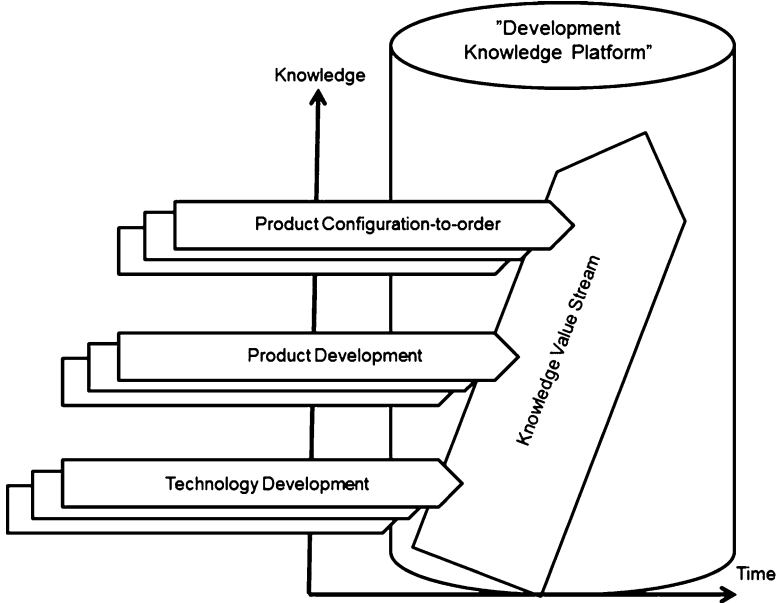


Fig. 5.4 Development processes and the *Development Knowledge Platform*

According to Mahmoud-Jouini and Lenfle (2010) the benefits from adopting a platform approach are such that the question no longer is whether to invest in a platform or not, but how to design it. The reusable knowledge gained in the different development processes is the basis for establishing a platform for further development, i.e., the *Development Knowledge Platform* in Fig. 5.4. How such a platform should be designed depends on the business environment of the individual company, its role in that environment and its product offerings. These prerequisites can vary substantially between, for example, an OEM company in the automotive business, which is mass-producing complex system products for the private consumer market, and a sub-supplier in the aerospace business providing low volumes of extremely customized high-tech subsystem solutions to a very limited number of highly qualified industrial customers. A concept for a platform aimed for the latter category has been proposed in research collaboration between WQL Chalmers and an aerospace sub-supplier company. The company is a supplier of subsystems to the aeroengine manufacturers with whom they perform collaborative product development.

The case company offers extremely customized subsystem solutions in low volumes to industrial customers that control the overall product architecture. A consequence of this is that reuse of ready designed components or modules, with fixed interfaces, is not an option. One customer would not accept direct carryover of a solution used by a competitor. The ambition is therefore instead to achieve scale benefits by reuse of more generic assets like adaptable system design concepts and

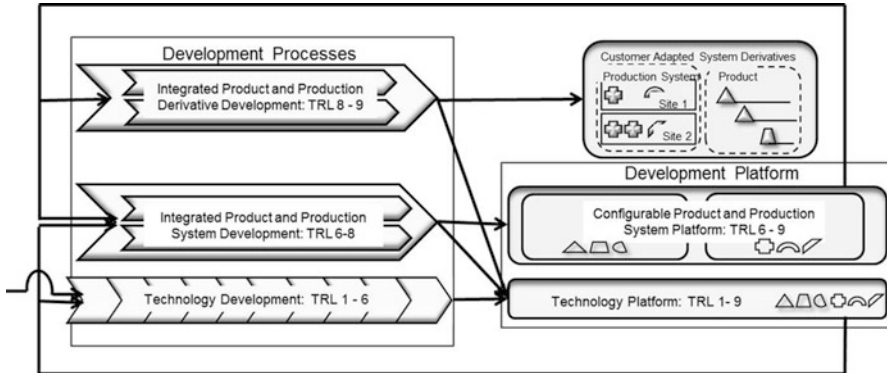


Fig. 5.5 Development processes and development concept platform

different kinds of methodologies and technologies mastered by the company (Berglund et al. 2008; Högman et al. 2009). The concept development platform therefore consists of:

- A technology platform, where product technologies, manufacturing technologies, design methodologies, manufacturing methodologies, supporting IT-based tools, and other product and production matters of interest, which have been explored to different maturity levels, are stored and managed for further product development support.
- An integrated product and production system family platform from which product and production system variants can be efficiently and effectively adapted within a defined bandwidth.

In Fig. 5.5, the development platform (lower right) and its related development processes (left) can be seen. The upper gray part symbolizes a system derivative resulting from a derivative development project where the platform has been used. Technology development process projects are initiated in order to fulfill experienced needs for change and improvement. These could, for example, be foreseen in existing product plans, triggered by competitors' product launches, or driven by new technological breakthroughs. Figure 5.5 shows that input to technology development (entering to the left in the process model) can be both new external knowledge and knowledge already existing in the platform. Note that the technology development process model symbol intends to show that many different technology development projects can be executed simultaneously. The result from a technology development project, i.e., new enhanced knowledge on an increased level of maturity compared to the input knowledge, is fed to the technology platform for future reuse.

The input to an integrated product and production system development process project can come both from the technology platform and the product and production system platform depending on the aim of the specific system development project. This could, for example, be to make use of new technology in an existing or new

system platform or to further develop it to extend its bandwidth without introducing new technology. The result is to be fed to both the technology and the product and production system platforms (although maybe in different versions). It is a contribution to a new system platform or an upgrading of an already existing system platform in terms of either bandwidth, maturity, or both.

Input to a system derivative development process project comes from the configurable product and production system platform together with system variant defining input parameter values. The aim of this process is to effectively and efficiently create a system derivative solution fulfilling specific functional as well as nonfunctional requirements within the bandwidth of the platform system. If the request falls within the bandwidth, this should be a matter of pure configuration. If not, the system descriptions in the platform are reused and adapted (engineered) in order to fulfill stated requirements. A prerequisite for this is platform systems descriptions, on acceptable to high maturity levels (TRL 6–9), that are configurable within specified bandwidths. The result of the derivative development process can be a proposed system solution to be given to a customer or to be used in different development activities of other stakeholders. It should also be fed to the technology platform as a highly mature system solution for potential future reuse.

## **5.4 Platform Descriptions and Platform System Models**

In this section, tentative approaches to describe and model the contents of technology and product and production system platforms, as described in Fig. 5.5, are discussed. The approaches have been developed in research collaboration with both OEM and sub-supplier companies active in the automotive and aerospace businesses.

### **5.4.1 Technology Platforms**

As stated above, a technology platform can contain product technologies, manufacturing technologies, design methods, manufacturing methods, supporting IT-based tools, and other product- and production-related matters of interest, which are important to reuse in technology as well as product and production system development. The maturity of different technologies managed in the technology platform can vary from very low to complete. The level of maturity is furthermore application dependent, meaning that the same technology can be completely verified and validated for one certain application, whereas it might be low for another where the consequences of its applicability are not yet fully understood. When describing the technologies in the platform, with the aim to support reuse, it is important not only to describe the technology as such but also all experiences made of its application so far. This means applications in different scenarios with

**Technology Platform**



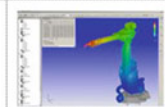

Last modified by Administrator 2012/02/11 10:26 [Comments \(0\)](#) · [Annotations \(0\)](#) · [Attachments \(5\)](#) · [History](#) · [Information](#)

### Purpose

The Technology Platform lays the foundation for the product and production platforms by collecting and displaying information about technologies underlying designs and processes used at the company. Experts, reports and previous projects dealing with the technologies can be found through the links from each page.

### Technologies

The company supports a wide range of technologies. They are divided in the following main groups:

			
<a href="#">Design Solutions</a>	<a href="#">Engineering Methods</a>	<a href="#">Manufacturing Methods</a>	<a href="#">Test &amp; Control Methods</a>

Tags: [+] Created by Administrator 2012/01/20 09:30

**Done**

Fig. 5.6 Technology platform portal (redrawn from screenshot)

references to existing products, previous projects, in-house experts, and PLM repositories. With this information available, a potential user of a certain technology will have access to all previous experiences made of that technology in the company. This enables him/her to better understand the technology and its limitations in different scenarios and also what new knowledge that might be needed and what adaptations that have to be made in his/her new application.

A Wiki-based approach has been adopted to create a portal, or entrance, to the technology platform (Bergsjö 2011; Corin Stig and Bergsjö 2011). Anyone involved in development activities within the company has access to this portal. This means that anyone can check in and have access to technology information on a first general or basic level. Here you can read what is presented and you can also make comments, ask questions, and make propositions regarding the described technologies. Referenced information, e.g., to certain projects, customer applications, or certain parts of the PLM repository, that is more or less classified will require certain access codes which are given to authorized users.

The technologies in the platform are divided into four categories: design solutions, engineering methods, manufacturing methods, and test and control methods as shown in Fig. 5.6. Within each category, each technology is described on an overall level on electronic *A3-sheets* following the Lean product development approach (see Fig. 5.7).

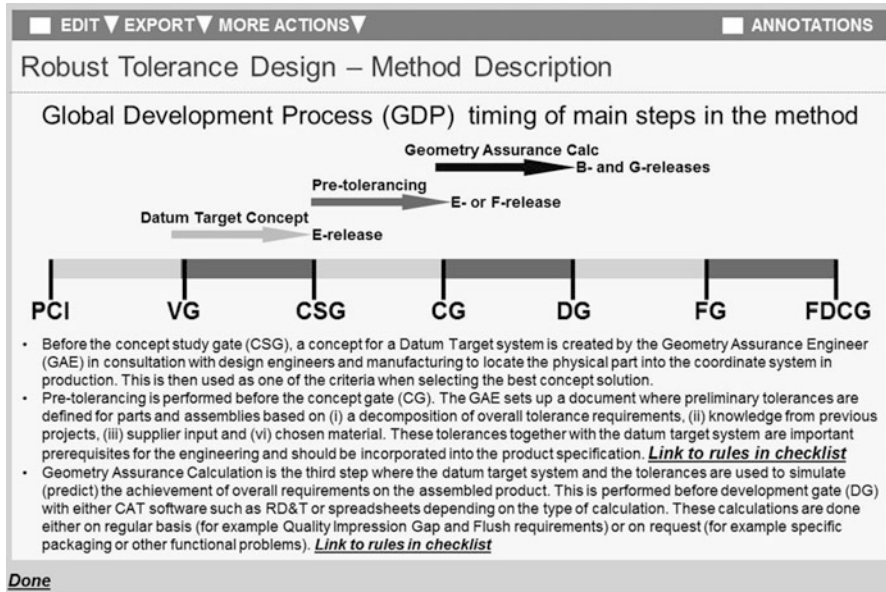


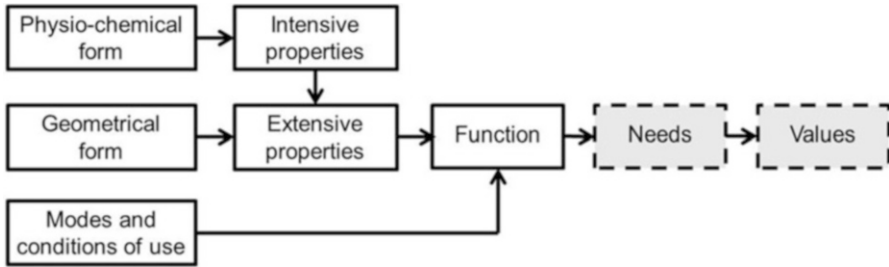
Fig. 5.7 Example of a robust tolerance design technology A3-sheet (redrawn from screenshot)

### 5.4.2 Product and Production System Platforms

As stated above, a product and production system platform is an integrated system family platform from which product and production system variants can be efficiently and effectively adapted within a defined bandwidth. It is here assumed that the products that are to be developed, as well as the production facilities that are to materialize them, are multi-technological systems consisting of different kinds of hardware (e.g., mechanical, fluid, electronic) and software subsystems. All systems are interacting with each other and the surrounding environment in different ways during the different phases of their life cycles.

The main aim of this kind of a platform is that it should be a sufficiently information-rich and adaptable knowledge source that will enable effective and efficient generation of quality assured system variants within certain constraining limits or bounds—the *bandwidth*. The bounds could be of both functional (related to what the system and its parts can do) and nonfunctional nature (related to what the system and its parts shall be). The systems described in the platform should be developed to a maturity level so that they could be reused:

- For new development of platform systems aimed for original or new settings
- For extension of original platform bandwidth in engineer-to-order settings
- For ordered configuration of quality assured variants within the platform bandwidth



**Fig. 5.8** Function as a combination of extensive properties (e.g., weight) and mode and condition of use, whereas extensive properties result from geometrical form and intensive properties (e.g., material density) which in turn are consequences of so-called physiochemical form

In order to handle these different requirements, a more abstract approach to describe the elements of the platforms has been chosen. The description, where the platform elements are modeled as autonomous, generic, configurable systems called *configurable components* (CC), was proposed by Claesson (2006). The approach is based on systems theory principles (Hitchins 2003) and design theory (Andreasen 1998; Hubka 1997). A system description, which is a model of a whole system family, contains information about both the system solution itself, the means to compose system variants and also its underlying requirements and motivations, i.e., its design rationale.

#### 5.4.2.1 Theoretical Background

In general, designing comprises a number of different activities, where each activity contributes to the creation of design information. The activities include:

- Defining the design problem to be solved
- Finding a design solution to the problem
- Designing a production process that can materialize the design as parts

According to Roozenburg and Eekels (1995) the core of designing is the transition of a functional description of a product to the description of its form. This can be expressed when it is known what functions the design should deliver. Normally, neither the design nor how it is to be used is known. Figure 5.8 can exemplify the situation with knowledge of wished functionality and absence of the design and how to use it. Here, the design process goes from right to left (in other words, in the opposite direction of the arrows). The reasoning from function to form is reasoning where very little is given and very much is asked. It is an open process that allows for many good solutions. It is also inherently non-deductive; it *cannot be grasped in an equation*. The designer's skills and knowledge play a central role during the reasoning. The result of this reasoning can be described as design

decisions documented in a product model. Deduction is the opposite where some kind of formula, which can be combined with some premises to find the solution, might be given.

Different design projects have different design goals, and different design approaches use different modes of reasoning. A supportive design model should support them all. A *configurable component*-based product platform model has the potential to provide much of this support.

Two technical system characteristics are of special interest:

- Synergy, often expressed as *the whole is greater than the sum of the parts*, which implies that the complete behavior of a system cannot be predicted by the behavior of its parts.
- Encapsulation—a system is delimited by its system boundary, with which the surrounding environment interacts, both through stimuli and response.

Hubka and Eder (1988) presented the theory of technical systems (TTS). At its core, it aims to provide a comprehensive theory to classify and categorize the information of technical systems into ordered sets of statements. For the proposed platform element description approach, the TTS concept of transformation and the different domains used to represent different aspects of a technical system are especially interesting. Technical systems are the principal means by which a transformation is achieved. The technical system exists only to realize a transformation from input to output. A combination of input and the system's internal states define the system's output as well as which internal state it will adopt.

Hubka and Eder present five abstract models of technical systems: purpose, process structure, function structure, organ structure, and component structure. Andreasen (1998) then proposed the theory of domains (ToD), based on Hubka and Eder's theory of technical systems. The model consists of four views: process domain, function domain, organ domain, and the component domain. The original ToD has been modified over the years. The chromosome product model (Mortensen 1999) is a generic structure based on the theory of domains.

Models like those referred above, containing more information than just the resulting design, are fundamental in any product development that has reached at least some degree of complexity. In his thesis, Andersson (2003) presents how the design intent, the design rationale, and the design history are related. The three concepts are explained as follows:

- *Design intent* forms the underlying reason why a certain artifact exists, where the design intent can be seen as the relationship between the intended behavior and the structure designed to realize the intended behavior, thus answering the question *why things are*.
- *Design rationale* includes not only the reason behind a design decision (the intent), but also the justifications for the decision, the other alternatives considered, the trade-offs evaluated, and the argumentation that led to that decision (Lee 1997). Thus, design rationale, in a post-decisional perspective, answers the question, *why things are, and why things are the way they are*.



- *Design history* also includes, besides design intent and rationale, the recorded process of the design process, describing *...how the artifact came into being*. This includes, among other things, the resources allocated and the actual progress of the design.

The function-means method is a systematic way of finding design solutions that fulfill functional requirements. The function-means model (or function-means tree, F-M tree) describes both requirements and solutions, and it has the advantage of being supported by a systematic design approach. A function-means model is a hierarchical model of one particular system, which is decomposed in subordinate subsystems.

The F-M tree is also a representation of Hubka's law, which states that "The primary functions of a machine system are supported by a hierarchy of subordinate functions, which are determined by the chosen means (organs)." The model that has evolved over time was originally developed by Tjalve (1976) and Andreassen (1980). In an analogy to the F-M tree, axiomatic design (Suh 1990) describes the zigzagging between functional requirements (FR) and design parameters (DP). The zigzagging points out the fact that a requirement cannot be decomposed into other requirements without identifying intermediate solutions (i.e., DPs).

Schachinger and Johannesson (2000) enhanced the function-means tree method by adding the abilities to describe more types of relationships and to separate functional requirements from nonfunctional requirements, constraints (C) as illustrated in Fig. 5.9. Function-means trees for a jet engine subsystem are shown in Fig. 5.15 (without constraints and linked external models/documents).

In this research the enhanced function-means tree is used as a backbone in a system's design rationale. This is in turn seen as a formalized description of a specification of a technical system. In axiomatic design terminology this description exists in the functional and physical domains. It is therefore assumed that its function domain parts (FRs and Cs) also can be mapped on its customer needs counterparts in the customer domain. How that mapping is achieved is outside the scope of this research.

Functional requirements are here defined as what a product, or an element of a product, actively or passively shall do in order to contribute to a certain purpose by creating internal or external effects. In this sense, they motivate the downright existence of a specific solution. The means, organs or design solutions<sup>1</sup> (DS), are the physical (e.g., components or features) or nonphysical (e.g., service or software) entities that can possibly fulfill a specific functional requirement. The role of the nonfunctional requirements (referred to here as constraints) is to delimit the allowed solution space for the functional requirements-driven design solutions. In contrast to functional requirements, constraints do not have specifically allocated design solutions.

---

<sup>1</sup> Means are renamed from "design parameter" to "design solution." One reason is to make the word "parameter" free, usable in parameterized designs.

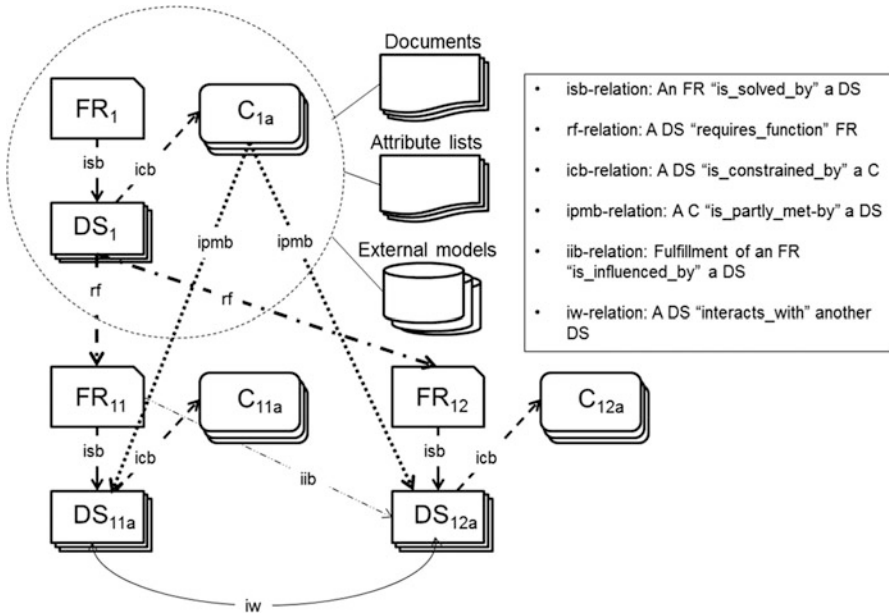


Fig. 5.9 Extended function-means tree

An organism is an example of an entity where the whole is more than the sum of its parts. More specifically speaking, the behavior of a system depends on its subsystems and their interactions. This additional contribution, the so-called emergent properties (Checkland 1981), cannot be attributed to any specific part of the system. Rather, they emerge only when the system as a whole is considered. Hitchins (2003) expresses this clearly when he states:

“The properties, capabilities, and behaviors of a system derive from its parts, from interactions between those parts, and from interactions with other systems.”

A key factor in successfully handling complex systems is to sometimes deal with a limited number of a system’s parts at a time, instead of with the complete system. Decomposition is a way of limiting the task. However, it has its drawbacks. Hitchins describes how decomposition will make the parts lose their interactions, thereby losing their context. Elaboration and encapsulation, on the other hand, look at the parts in situ and *in context*, thus maintaining their interactions. This can be exemplified with IDEF0-modeling where an overall system encapsulates subsystems which in turn encapsulate their subsystems and so on. Elaboration then means moving down in such a hierarchy focusing more details. Encapsulation can be compared to moving up in the hierarchy while placing containers around sets of entities. The container does not cut the interactions, as decomposition does. Instead, they remain intact. Encapsulation conceals unnecessary details and reduces the perceived complexity of the system, as shown in Fig. 5.10. The elaboration-encapsulation concept is central in the configurable component modeling approach which is described in the next section.

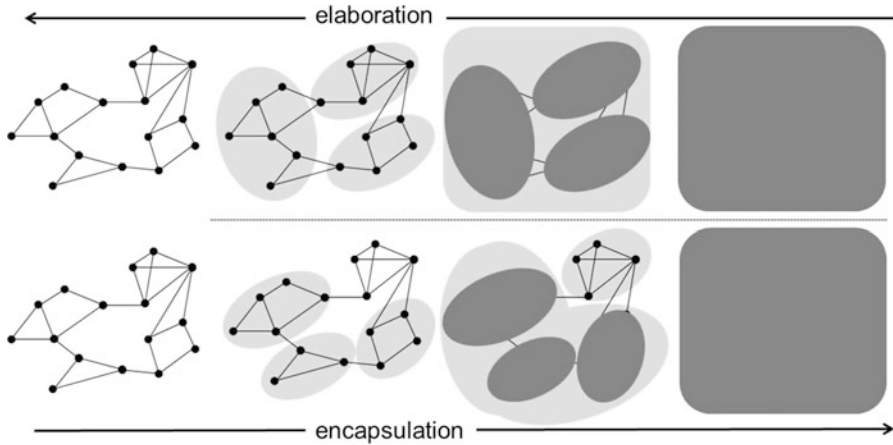


Fig. 5.10 Two alternative elaborations and encapsulations (Hitchins 2003)

#### 5.4.2.2 The Configurable Component Concept

A design can be totally described by either one configurable component or one configurable component composed using other configurable components, which in turn may be composed using other configurable components and so on. The latter scenario can support a decentralized organizational setup where a designer is responsible for his/her level of granularity of the design within an organization.

A configurable component needs to be referred to by multiple different compositions who are using it for realization of a system instance. Such references may cross product as well as company boundaries. This requires that each configurable component should be as self-contained as possible. It also requires that parameters that describe variability should be expressed in a terminology relevant for the configurable component itself.

A number of successive compositions can be perceived as a structure. For such a structure to exist, there must be a method capable of traversing a number of configurable components and presenting them as a structure. Such a traverse may start with any configurable component.

The configurable component has some constructs that can be regarded as basic—encapsulation, variability, and composition. Encapsulation will conceal the internal structure of the configurable component for the outside environment. Access is gained through interfaces, which make selected parts of the internal content available. Variability (in other words, the ability to represent a number of similar variants of a design or system with one configurable component) is achieved by parameters. Different types of parameters are used to describe different aspects of the variability: design parameters, performance parameters, and variant parameters. Design parameters are controlled directly by the designer, whereas performance parameters are a consequence of the design. Finally, variant parameters are used as

a convenient way of defining sets of design parameter values. A specific variant, a configuration, is achieved by giving values to the parameters of the configurable component. Composition is the ability of a configurable component to compose itself using other systems (also modeled as configurable components) to fulfill its purpose. Because both the using (parent) system and the used (child) systems are configurable and the child will be dependent on the parent's configuration, methods are needed to select the correct configurations.

A configurable component is an autonomous configurable system family model of encapsulated combined subsystem families with different purposes (i.e., it has multi-functionality). A product platform, carrying a number of product variants, can thus be achieved by a number of such configurable components. The different members (or possible system variants) that can be carried by the platform are determined by the bandwidths (variation ranges) of the involved configurable components.

Claesson (2006) made the first coherent presentation of the configurable component concept. It was created to deal with the automotive industry's challenge of combining customer demands of individualized products with company demands of economy of scale. The need for a more capable product description that could be used by all engineering disciplines, in cross-functional teamwork during the complete product life cycle, was also identified and addressed. The approach was first applied within the Saab Automobile Company when the second generation of the Saab 9-3 was developed and produced.

In Fig. 5.11 the basic constructs and internal resources encapsulated in a configurable component are illustrated (Gedell et al. 2008; Johannesson and Gedell 2009; Gedell 2011). The system family, represented by the CC, is described by its design rationale (DR) with its governing functional requirements (FR), nonfunctional requirements [constraints (C)], and linked design solutions (DS) in enhanced function-means tree structures. Interactions (IA) and interfaces (I/F) are special design solutions handling the systems interactions with its environment. All internal objects are parameterized and associated with parameters and parameter values. They are coupled to variant parameters (VPs) in the control interface (CI) and design rules in order to achieve variability (*bandwidth*). Composition methods (rules), described within the composition set (CS), enable the CC to use other CCs and instantiate composition elements (CEs) that will correspond to its design solutions (DSs) prescribed in the design rationale (DR).

In order for a configurable component to benefit from the function-mean model's advantages, the function-means model must be adapted and applied in such a way that it can represent multiple similar designs realized by combined systems with different purposes. It must also be extended with methods to relate the combined systems' variability ranges (bandwidths) to each other as constraints in a using (parent) CC will be imposed on a used (child) CC.

For each required function, a design solution with required sub-requirements and corresponding sub-solutions [i.e., a function-means tree (here, a FR-DS structure)] is developed. All such FR-DS structures, enhanced with decomposed system constraints, together constitute the design rationale (DR) of the system modeled

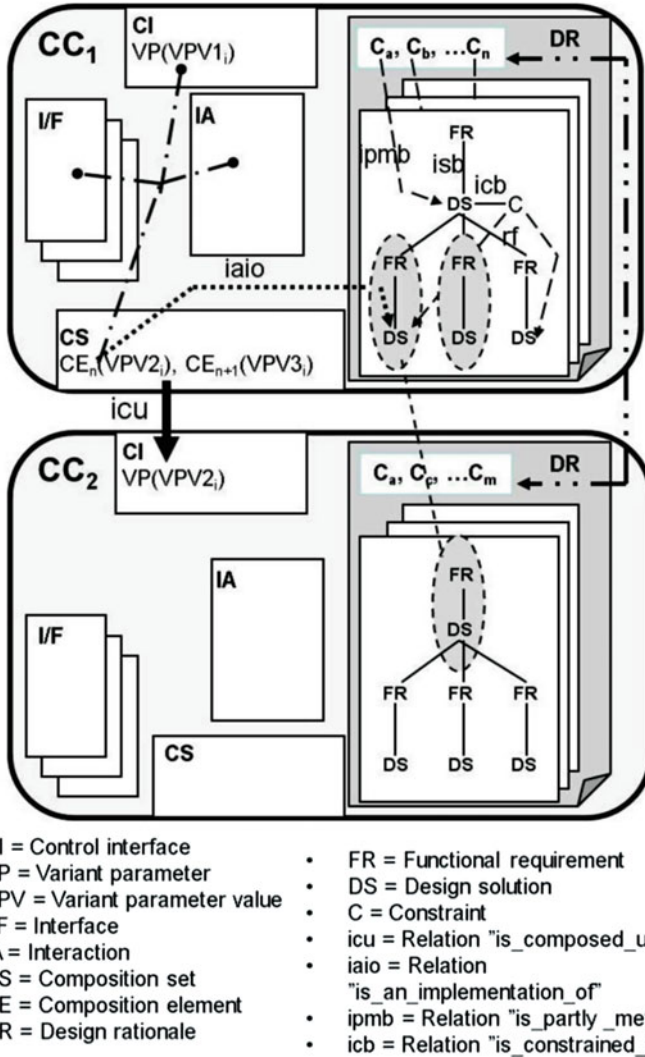


Fig. 5.11 A configurable component and its composition

as a configurable component. Instead of being a function-means tree with one FR-root, the DR is seen as a collection of embedded function-means models with multiple starting points. The top design solution, which is the configurable component itself, represents the complete design (see Fig. 5.11).

Sub-requirements and sub-solutions are useful when designers, developing a system platform, are to define a configurable component's composition (in other words, to identify other configurable components to be used in a composition). If a set of sub-requirements and sub-solutions match corresponding items in the

DR of an existing CC (see Fig. 5.11), it is a perfect choice. An *icu*-relation (*is\_composed\_using*) between the using (parent) CC and the used (child) CC will then be created. As a result the using CC will be able to transfer instantiated variant parameter values (VPVs) to the used CC for its variant instantiation. A configured variant of the used CC is then available as a composition element (CE) in the using CC's composition set (CS). If no such candidate can be found, the identified sub-requirements and sub-solutions can constitute a plea for further or new development of a CC-modeled system family to be used.

## 5.5 A Platform Life Cycle Management Software Environment

For the purpose of building CC models of configurable system families a software tool, Configurable Component Modeler (CCM) (Edholm et al. 2009, 2010a), has been created. The underlying idea is that CCM shall be an aiding tool for system family design including both system modeling and knowledge retrieval. It shall also act as host for the system family descriptions defining a CC-based product and production system platform. All the CC features are to be modeled, and the three reuse scenarios indicated in Sect. 5.4.2, new development, bandwidth extension, and variant configuration, are to be supported. CCM can be described as a platform modeling and configuration (PMC) system which is to be a part of a PLM environment together with PDM, CAD/CAE, RM, and other software tools in a service-oriented architecture (SOA) (Wahl et al. 2010; Levandowski et al. 2011a) as shown in Fig. 5.12.

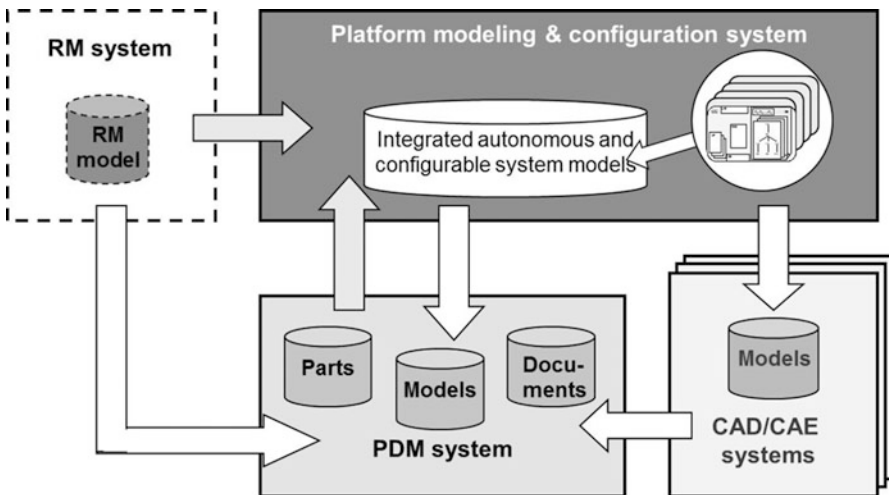


Fig. 5.12 Platform PLM architecture

With the chosen approach a complete product platform description will be carried by a number of software tools integrated in a PLM environment, e.g.:

1. A platform modeling and configuration (PMC) system
2. A PDM system
3. CAE systems like CAD, FEM, and MBS
4. A requirement management (RM) system

The PMC and the PDM systems are mandatory parts of the architecture. The PMC system, which is the user interface system for product variant configuration as well as for platform design and development, contains the configurable CC system family models defining a platform. In order to fully define a configured instance, the CC models need to link relevant instance related part models and documents managed in the PDM system to their contained objects and relations. This is done with methods linked to the objects and relations in the CC models. The CCM software provides the ability to create such methods that facilitate communication with other software systems.

The role of the PDM system is to be a carrier and manager of the parts, documents, and information belonging to the product platform. It is furthermore a carrier and manager of the information identifying all product models in different PLM systems belonging to the platform. Those are, for example, CAD models, FEM models, and MBS models but also RM and PMC models. Finally the PDM system has its role as process work-flow manager.

The design rationale models in the CC structures are carrying the knowledge about the origin of each CC. They explain what the subsystems configured by the CCs should do and be, how the solutions are realized, and why they are realized the way they are. The functional requirements (FRs) and constraints (Cs) contained in these models originate from the product platform specifications. If these are available as RM models in requirement management systems, appropriate links should be established between the RM models and the PMC models for traceability reasons. Such links could be realized by using CCM methods linking FR and C objects in the CC models to corresponding requirement items in the RM system models. So far no such RM models have been available, and no such links have been established. The necessary input to the created design rationale models has been elicited from requirement documents and interviews with experienced engineering designers.

In order to configure product instances that contain, for example, hardware components that are configured in size and shape, the CCs in the PMC system must refer to CAE systems containing configurable hardware models. Parameterized CAD models are such examples. Here generic geometry models, with rules governing the geometric configuration, are stored in CAD system database. The administration of the CAD models is handled by the PDM system. By using a method, a CC object in the PMC system can identify a model in the PDM system and call it in the CAD system. The instantiated parameters that have been generated in the PMC system and govern the configuration of the parametric CAD model are then transferred to the CAD system, and the parametric CAD model is instantiated. In a similar manner other CAE systems, used for analysis and simulation of product properties, are linked to the PMC and PDM systems.

## 5.6 An Industrial Case Study

An exploratory case study has been ongoing at WQL Chalmers together with an aeroengine subsystem supplier since 2007 (Levandowski et al. 2011b). The company supplies subsystems to all big aeroengine manufacturers worldwide. Examples of such systems are turbine exhaust cases (TEC) as shown in Fig. 5.13. A TEC is a subsystem found in the rare of jet aeroengines. It has two main functions, to guide the flow of the hot exhaust gases and to provide the attachments needed to mount the engine to the airplane wing structure. It is subjected to very high structural as well as thermal loads.

Historically structures like the TEC have been manufactured as castings. These have been highly customized and designed more or less from scratch for each customer. The possibility to reuse previous results has been very limited, and the development lead times have been long. A new concept, *fabricated structures*, has been adopted in order to increase reuse possibilities and shorten lead times. With this concept a TEC is a welded assembly of components, and in this case study, a product platform in the form of a configurable system family has been created for fabricated TECs. The CC-based approach described above has been adopted for this purpose. For realization of product variants, the product platform makes use of modeling, analysis, and manufacturing methods retrieved from a technology platform organized as described previously.

The TEC family system is defined by a TEC CC, a guide vane CC and an inner ring CC. Instances of the three are shown in Fig. 5.14. The TEC CC uses the vane CC and the inner ring CC to compose system variants of itself.

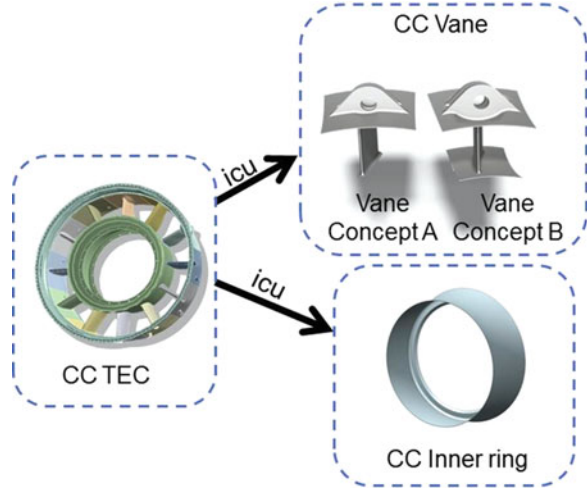
The principal design rationales (DRs) for the three CCs are shown without modeled constraints (Cs) and linked documents and models to the different DR objects (as shown in Fig. 5.9) in Fig. 5.15. Note that the TEC DR has two FR roots. The different functional requirements (FRs) on the TEC system and its parts, as well as the design solutions (DSs) derived to fulfill these requirements, are modeled



**Fig. 5.13** Turbine exhaust case—TEC



**Fig. 5.14** TEC system CC images with instance examples (relation  $icu=is\_composed\_using$ )



as objects in function-means structure models in the DRs. A gray arrow from an FR to a DS represents an *is\_solved\_by* relation, and a black arrow from a DS to an FR a *requires\_function* relation.

Functional requirement-related (FR) properties are gas flow rate  $Q$  and engine mass  $m$ . Nonfunctional requirement or constraints related properties are gas temperature  $T$  and gas pressure  $p$ . Design solution-related (DS) characteristics, or solution defining parameters, are number of vanes  $N$ , type of vanes  $A/B$ , and TEC diameters  $D_{outer}/D_{inner}$ . The bandwidth of the modeled TEC system family is defined by the allowed variation ranges of these requirement-related properties and design solution-related characteristics.

Within the bandwidth the TEC system platform should be able to provide valid TEC variant solutions of different dimensions, vane structures, and number of vanes. To explore this, as well as for the purposes of exploring knowledge retrieval for new platform development and bandwidth extension, a TEC platform PLM software architecture, shown in Fig. 5.16, is built in the referred case study (Levandowski et al. 2011b).

The CCM software plays the role as PMC system and the commercial software *shareSpace* as PDM system in the TEC platform, as described in the previous chapter. For geometric modeling, and for hosting generic parameterized models defining the geometries of the design solutions (DSs) in the design rationales, the CAD system Siemens NX7 is used.

As this platform has platform elements with configurable geometric interfaces, the geometric robustness and tolerance allocation are optimized for each interface instance. For this purpose, the engineering method *Robust Tolerance Design* (Edholm et al. 2010b) has been adopted from the technology platform. The commercial software RD&T is used to execute the method. Other engineering

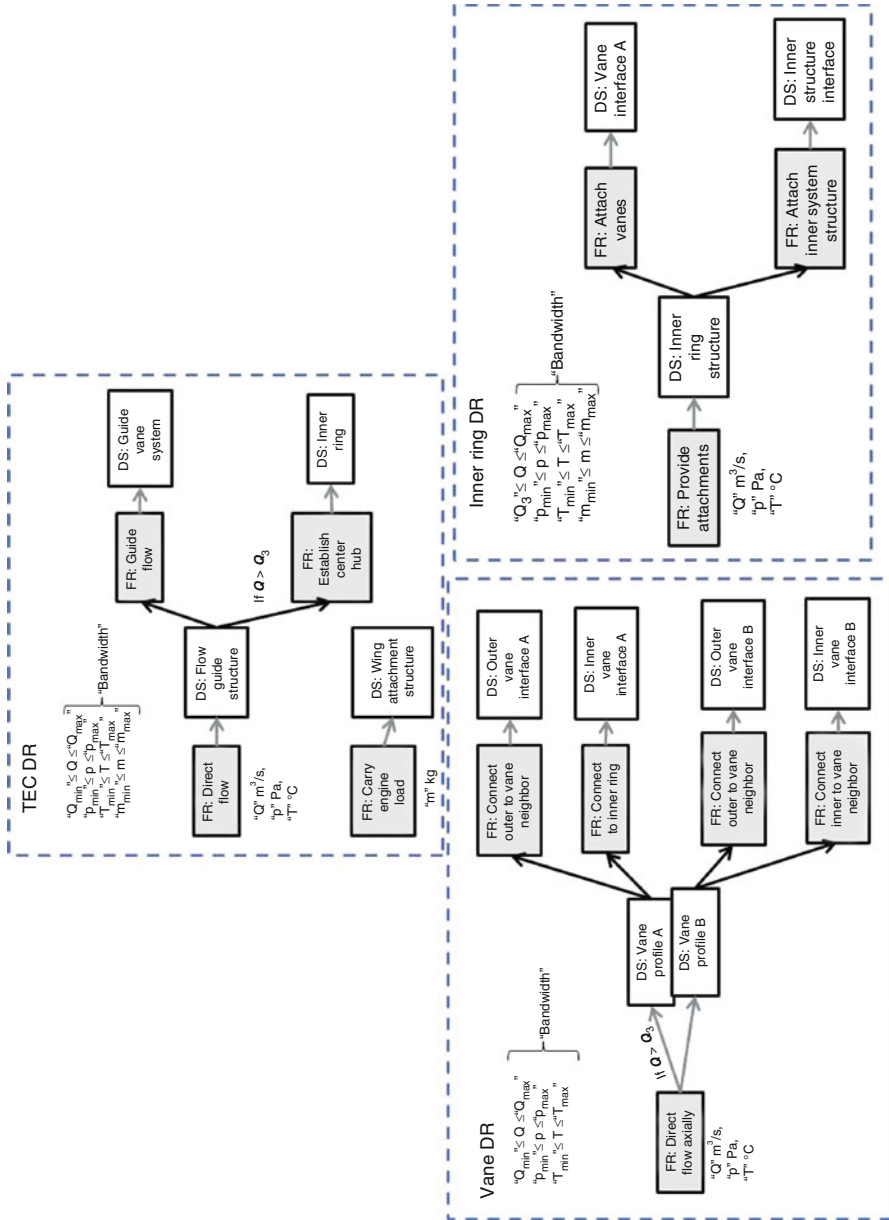


Fig. 5.15 TEC system design rationales

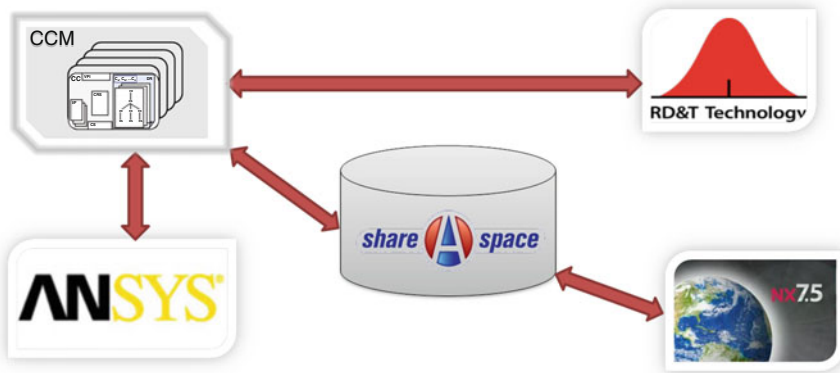


Fig. 5.16 TEC platform system software architecture

methods, adopted from the technology platform and used for analysis and simulation of TEC instance properties, are methods for analysis of pressure loss, buckling load, thermal stress, and shear compliance. These analyses are executed by the commercial software system *ANSYS Workbench*.

In the presented state the TEC platform can be used for TEC variant configuration governed by solution parameters. Parameter values specified within the bandwidth will result in a configured variant ready for property analysis with the described methods. Comparisons between different variants' properties, and choice of *best variant within the design solution bandwidth*, can be made. Specification of input parameter values outside the bandwidth will result in a message that the input values are invalid and that no TEC variant can be configured.

Ongoing development is focused on implementing the possibility to use functional and nonfunctional property values, *requirement parameter values*, as input to variant configuration in CCM. With this in place it will be possible to specify variants to be configured in terms of required properties. An available design solution space, defined by a *requirement fulfillment bandwidth*, can then be explored in order to find functionally valid solutions. It can also be used to identify needs for bandwidth extension or redesign of the platform based on reuse of knowledge contained in the existing platform system family CC models.

## 5.7 Conclusions

The view put forward by Robertson and Ulrich (1998) of a platform as a source for reuse of assets in a wide sense, not limited to a collection of common parts, has been adopted in the research at WQL Chalmers. With this approach to platforms, it is possible to address a wide range of needs appearing in different industrial settings.

Mandatory needs like simultaneous economies of scale and customization benefits, motivating most platform efforts, can of course be met with this approach. In addition other needs like development support across all phases of the platform life cycle and collaboration support in the supply chain can also be fulfilled.

Development support across life cycle phases is enabled by letting platforms contain information and knowledge of different levels of maturity. This can be achieved by using TRL classification of technology platform elements in the technology platform as well as of alternative design solutions (DSs) in the product and production system platform elements. A platform can then provide support from early conceptual development stages (low TRL levels) when potential solutions should be allowed to be incomplete, incompatible, and inconsistent to the operative product variant configuration stage (TRL 9) when the reused information must be complete, compatible, and consistent (Gedell et al. 2011). An important key to supply-chain collaboration support is the use of generic, autonomous, and configurable system families as platform elements. This makes it possible for a supplier to rapidly respond to a customers' request by means of system variant configuration or an effective and efficient re-engineering effort (Wahl et al. 2010).

More abstract, generic, and configurable platform elements counteract platform degradation as the concept as such, with its modular architecture, effectively enables addition of system families and system family bandwidth extensions. A consequence of this could be that a platform of this kind would never be obsolete as it is continually upgraded with new technologies and new system family solutions. It might live forever as a key knowledge source for product and production system development in a manufacturing company. The research on platform based product and product system development at WQL Chalmers continues along these lines.

**Acknowledgments** The presented research is carried out at the Wingquist Laboratory VINN Excellence Centre within the Area of Advance – Production at Chalmers, supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA). The support is gratefully acknowledged.

## Appendix

Acronym	Complete wording	Explanation
BOM	Bill of material	A listing of parts/components/subsystems contained in a system product
C	Constraint	A requirement limiting the design solution space
CCM	Configurable Component Modeler	A software for modeling CCs and CC structures
CC	Configurable component	A configurable system family model

(continued)

(continued)

Acronym	Complete wording	Explanation
CE	Composition element	An object within a CC enabling the CC to compose a variant of itself by configuring and using (incorporating) a variant of another CC
CI	Control interface	The interface used to give input (set variant parameter values) to a CC
CS	Composition set	The set of composition elements within a CC
DR	Design rationale	A description of a design solution and its governing requirements
DS	Design solution	A design concept fulfilling one (and only one) specific governing functional requirement
F-M tree	Function-means tree	A hierarchical integrated function and design solution model
FR	Functional requirement	A requirement specifying a wanted behavior of a product
IA	Interaction	An interaction between two interacting technical systems
I/F	Interface	A feature in a technical system interacting with its counterpart in another technical system
icb	is constrained by	A causal relation between (from-to) a DS and a C
iib	is influenced by	A secondary causal relation between (from-to) an FR and a DS
ipmb	is partly met by	A causal relation between (from-to) a C and a DS
isb	is solved by	A primary causal relation between (from-to) an FR and a DS
iw	interacts with	A causal relation between two different DSs
PMC	Platform modeling and configuration	
rf	requires function	A causal relation between (from-to) a DS and an FR
TEC	Turbine exhaust case	A subsystem in an aeroengine
ToD	Theory of domains	
TRL	Technology readiness level	A measure of maturity for use of technical solutions
VP	Variant parameter	A parameter used to specify a specific system variant
VPV	Variant parameter value	
TTS	Theory of technical systems	

## References

- Andersson F (2003) The dynamics of requirements and product concept management. Dissertation, Chalmers University of Technology
- Andreasen MM (1980) Syntesemetoder på Systemgrundlag – Bidrag till konstruktionsteori (in Danish). Dissertation, Lund University
- Andreasen MM (1998) The theory of domains. In: Proceedings of the workshop understanding function and function-to-form evolution, Cambridge
- Baldwin CY, Clark KB (2000) Design rules: the power of modularity. MIT Press, Cambridge, MA
- Berglund F, Bergsjö D, Högman U, Khadke K (2008) Platform strategies for a supplier in the aircraft engine industry. ASME DETC2008-49526

- Bergsjö D (2011) Process and IT support for technology platform development and use. In: Proceedings of IAMOT 2011, Miami Beach, FL
- Checkland P (1981) Systems thinking, systems practice. Wiley, Chichester
- Claesson A (2006) A configurable component framework supporting platform-based product development. Dissertation, Chalmers University of Technology
- Corin Stig D, Bergsjö D (2011) Means for internal knowledge reuse in pre-development – the technology platform approach. In: Proceedings of ICED 2011, Copenhagen, Denmark
- Edholm P, Lindquist Wahl A, Johannesson H, Söderberg R (2009) Knowledge-based configuration of integrated product and process platforms. ASME DETC2009-86540
- Edholm P, Levandowski C, Johannesson H, Söderberg R (2010a) Applied CC configuration in PDM/CAD environment. In: Proceedings of INTECH 2010, Prague, Czech Republic
- Edholm P, Johannesson H, Söderberg R (2010b) Geometry interactions in configurable platform models. In: Proceedings of design 2010, Dubrovnik, Croatia, pp 195–204. ISBN/ISSN: 978-953-7738-07-05
- Erens FJ (1996) The synthesis of variety: developing product families. Dissertation, Eindhoven University of Technology
- Ericsson A, Erixon G (1999) Controlling design variants: modular product platforms. Society of Manufacturing Engineers, Dearborn, MI
- Gedell S (2011) Efficient means for platform-based development – emphasizing integrated information-rich system models. Dissertation, Chalmers University of Technology
- Gedell S, Johannesson H, Holmberg L (2008) Design rationale for efficient product platform development – a systematic configurable component approach. In: Proceedings of TMCE 2008, Izmir, Turkey
- Gedell S, Claesson A, Johannesson H (2011) Integrated product and production model – issues on completeness, consistency and compatibility. In: Proceedings of ICED 2011, Copenhagen, Denmark
- Gershenson JK, Khadke KN, Lai X (2006) A research roadmap for robust product family design. Department of Mechanical Engineering-Engineering Mechanics, Michigan Technological University, Houghton, MI
- Halman J, Hofer AP, van Vuuren W (2003) Platform-driven development of product families: linking theory with practice. *J Innov Manage* 20:149–162
- Hitchins DK (2003) Advanced systems – thinking, engineering, and management. Artech House, Norwood, MA
- Högman U, Bergsjö D, Anemo M, Persson H (2009) Exploring the potential of applying a platform formulation at supplier level – the case of Volvo Aero Corporation. In: Proceedings of ICED 2009, Stanford, CA, pp 227–238
- Hubka V (1997) Principles of engineering design. Heurista, Zurich
- Hubka V, Eder WE (1988) Theory of technical systems – a total concept theory for engineering design. Springer, Berlin
- Jiao JR, Simpson TW, Siddique Z (2006) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18(1):5–29
- Johannesson H, Gedell S (2009) Knowledge based configurable product platform models. In: Piller FT, Tseng MM (eds) Handbook of research in mass customization and personalization, vol 1. World Scientific, Singapore, pp 357–375
- Jose A, Tollenaere M (2005) Modular and platform methods for product family design: literature analysis. *J Intell Manuf* 16:371–390
- Kennedy MN, Minnock E, Harmon K (2008) Ready, set, dominate. The Oaklea, Richmond, VA
- Lee J (1997) Design rationale systems: understanding the issues. *IEEE Expert* 12(3):78–85
- Levandowski C, Edholm P, Ekstedt F, Carlson JS, Söderberg, Johannesson H (2011a) PLM architecture for optimization of geometrical interfaces in a product platform. ASME DETC11-47801

- Levandowski C, Corin Stig D, Bergsjö D, Forslund A, Högman U, Söderberg R, Johannesson H (2011b) An integrated approach to technology platform and product platform development. *Concurrent Engineering* March 2013 21:65–83, first published on December 12, 2012 doi:10.1177/1063293X12467808
- Mahmoud-Jouini SB, Lenfle S (2010) Platform re-use lessons from the automotive industry. *Int J Oper Prod Manage* 30(1):98–124
- Mankins JC (1995) Technology readiness levels: a white paper. Advanced Concepts Office, Office of Space Access and Technology, NASA
- Meyer MH, Lehnerd AP (1997) The power of product platforms: building value and cost leadership. The Free Press, NY
- Mortensen NH (1999) Design modeling in a designer's workbench – contribution to a design grammar. Dissertation, Technical University of Denmark
- Prencipe A (1998) Modular design and complex product systems: facts, promises and questions. *Complex Product Systems*, Publication No. 47
- Robertson D, Ulrich K (1998) Planning for product platforms. *Sloan Manage Rev* 39(4):19–31
- Roozenburg NFM, Eekels J (1995) Product design: fundamentals and methods. Wiley, Chichester
- Schachinger P, Johannesson H (2000) Computer modeling of design specifications. *J Eng Des* 11(4):317–329
- Simpson TW (1998) A concept exploration method for product family design. Dissertation, Georgia Institute of Technology
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf* 18(1):3–20
- Simpson TW, Maier JRA, Mistree F (2001) Product platform design. *Res Eng Des* 13(1):2–22
- Simpson TW, Siddique Z, Jiao RJ (2005) Product platform and product family design: methods and application. Springer, New York
- Suh NP (1990) The principles of design. Oxford University Press, New York
- Tjalve E (1976) Systematic design of industrial products – tools for the design engineer (in Danish). Akademisk Forlag, Copenhagen
- van Veen EA (1991) Modeling product structures by generic bills-of-material. Dissertation, Eindhoven University of Technology
- Wahl A, Gedell S, Johannesson H (2010) Supply-chain product development collaboration using configurable product platform models. ASME DETC2010-28014

# Chapter 6

## Quantifying the Relevance of Product Feature Classification in Product Family Design

Conrad S. Tucker

**Abstract** The methodology proposed in this chapter aims to address the link between the evolution of product feature relevance and the implications to product platform and product family design. By quantifying relevant/irrelevant product features to be included in next-generation product platform design, designers can identify the stand-alone or platform sharing components required to achieve desired product functionality. A data mining algorithm is introduced that uses time series data (consisting of product features) to determine the *standard*, *nonstandard*, and *obsolete* product features in the design of next-generation products. Product features are then mapped to engineering components/modules by employing data mining Natural Language Processing techniques that quantify the functionality requirements that are needed for a given set of product features. The goal of this work is to demonstrate the value of incorporating evolving product feature trends in the market space directly into product platform and product family sharing decisions.

### 6.1 Introduction

The increase in globalization and the prevalence of low-cost communication infrastructure present ever-increasing challenges for enterprise decision makers aiming to satisfy customer needs. In recent years, companies have had to consider the impacts of a socially connected digital age in shaping customer preferences and expectations in the market space (Tucker and Kim 2011a). The evolution of product preferences in the market space can be highly dynamic and difficult to capture using traditional customer-driven frameworks employed by design engineers. Mass customization has been proposed as a viable approach to accommodate the diverse

---

C.S. Tucker (✉)  
Industrial Engineering and Engineering Design, The Pennsylvania State University,  
University Park, PA, USA  
e-mail: [cst14@psu.edu](mailto:cst14@psu.edu)



product preferences in the market space. From an engineering perspective, however, mass customization presents the added challenge of establishing design and manufacturing processes to meet the needs of mass customization. Product family design is an enterprise-driven strategy aimed at mitigating the added costs that arise due to product customization. The two design strategies that have been proposed in the product family design literature are the *Bottom-Up* approach and the *Top-Down* approach (Simpson et al. 2001). Commonality indices proposed in the literature investigate component/module sharing strategies for existing products within a product family and are well suited for Bottom-Up product family design. In Top-Down product family design, a product family emerges from an existing market-driven need. The data mining component classification framework in this work will enable designers to identify components that are well suited for sharing in the product family design process through the use of large-scale market-driven product feature preference data.

## 6.2 Related Work

This section presents work relevant to the three main aspects of this research: (1) Data mining-driven product design, (2) Translating customer needs into engineering targets, and (3) Product platform and sharing decisions.

### 6.2.1 Data Mining-Driven Product Design

Data Mining-Driven Product Design is an emerging field of research aimed at incorporating large-scale data in the design of next-generation products (Braha 2001). Agard and Kusiak employ data mining association rules to cluster product functions in the design of product families (Agard and Kusiak 2004). Tucker and Kim (2008) employ Naive Bayes Classification techniques that enable designers to identify novel product feature combinations in a high dimensional product feature space. Moon et al. (2006) employ data mining Fuzzy c clustering techniques as a platform identification strategy in product family design. Data Mining techniques have been employed by Tucker and Kim to determine the optimal product feature combination for product family optimization (Tucker and Kim 2009; Tucker et al. 2010). Moon proposes a Data Mining framework for extracting design knowledge for product platform and variant design (Wang 2008).

While the aforementioned data mining techniques proposed in the literature aim to address product family design problems, they are static in nature and primarily consider large-scale data at an instant in time, hereby omitting the changes in product feature preferences that may occur in the market space over time. In order to accommodate evolving product trends in the market space, Tucker and Kim propose a temporal product feature classification algorithm that classifies

product features as *Standard*, *Nonstandard*, or *Obsolete*, based on their time series predictive power (Tucker and Kim 2011b). The classification of product components will enable design engineers to determine when to retire certain components (classified as obsolete in the metric), include in the design of a product platform (classified as standard in the metric), or aid in the creation of modules for product variants (classified as nonstandard in the metric).

### ***6.2.2 Translating Customer Needs into Engineering Targets***

Quality function deployment (QFD) is a well-established approach employed in the design community for translating customer preference requirements into engineering design targets/functional specifications (Pullmana et al. 2002). A house of quality (HOQ) would be designed, mapping the customer requirements into tangible engineering design targets (Bouchereau and Rowlands 2000). Customer preferences towards certain product features can be weighted through feature rankings acquired through surveys or focus groups (Kwong and Bai 2003). As a result, a QFD matrix can be used to depict the interdependence between customer requirements and the engineering metrics (EM).

The QFD model is highly dependent on the domain expert (engineers) translating the customer wants into engineering metrics. As the complexity of modern technology increases, so does the availability of product features and customization options. The increased product feature space (high dimensional feature space) and the highly dynamic nature of many consumer markets today make traditional translation of customer preferences into engineering metrics cumbersome. Furthermore, the expertise of these techniques is limited to the domain expert(s), making the process highly dependent on a subset of the product development team. By employing a data mining-driven approach to customer preference modeling and then translating the knowledge gained into tangible engineering metrics, design engineers will be able to incorporate market-driven trends during the translation of customer wants into engineering specifications. Instead of relying on survey or focus group feedback in an effort to quantify the evolution of product preferences in the market space, designers can employ the data mining methodology proposed in this chapter as a means of generating predictive models about evolving product feature preferences that can then be used for product family optimization.

### ***6.2.3 Product Platform and Sharing Decisions***

A *product platform* can be defined as a set of parameters/features or components that are shared across products within a product family (Simpson et al. 2001). Meyer and Lehnerd provide guidelines for product platform development and encourage companies to design products around a shared platform, rather than

subsequent independent designs (Meyer and Lehnerd 1997). *Commonality* refers to the level of sharing of components/subassemblies, processes, etc. across different products within a family of products (Boas 2008). Commonality therefore has the ability of reducing manufacturing and design costs (by sharing the same component across different products), while concurrently providing the level of product diversity expected within the market space. The trade-off between product commonality and product diversity has been studied extensively in the literature and is discussed in Simpson et al. (2001). de Weck highlights the challenges that exist in determining the extent of product platforming in product family design (Simpson et al. 2006).

Several commonality metrics have been proposed in the literature in an effort to quantify the effects of platform sharing decisions on product family design. For example, Collier proposed the degree of commonality index (DCI) as a way to measure the ratio of common components existing among products within a product family to the total number of components (Collier 1981). A modified version of the DCI called the total constant commonality index (TCCI) has absolute bounds (0–1), hereby making commonality comparisons within and between product families more quantifiable (Wacker and Trelevan 1986). The commonality index (CI) proposed by Martin and Ishii measures the ratio of unique components in a product family and the total components in a product family (Martin and Ishii 1996, 1997). The Percent Commonality Index (%C) measures product commonality within a shared product platform, rather than across product families using a weighted sum of multiple variables for a total commonality scale ranging from 0 (no commonality) to 100 (complete commonality) (Siddique et al. 1998). Another extension of the DCI called the component part commonality index (CI<sup>(C)</sup>) takes into account factors such as the cost of each component, product volume, and quantity per operation in determining the effects of component sharing decisions on a product family (Jiao and Tseng 2000). The product line commonality index (PCI) is a departure from traditional commonality indices that penalize broad product variation and instead, penalizes products with nonunique components within a product family (Kota et al. 2000). The generational variety index (GVI) proposed by Martin and Ishii measures the level of redesign work needed for future iterations of a product and helps designers determine which components may change over time (Martin and Ishii 2002). The comprehensive metric for commonality (CMC) is a data intensive approach to product commonality based on the components', size, geometry, material, manufacturing process, assembly, cost, and allowed diversity in the family (Thevenot and Simpson 2006). Alizon et al. (2009) propose a commonality diversity index (CDI) that compares components relating to a specific function(s) and investigates the trade-off between commonality and diversity based on the product family's functional requirements. With a plethora of commonality metrics proposed in the literature, Simpson et al. (2012) approach the product platforming problem by proposing an integrative approach that incorporates a market segmentation grid, the GVI, design structure matrix (DSM), commonality indices, mathematical modeling and optimization, along with multidimensional data visualization tools.

The methodology proposed in this chapter aims to address the link between the evolution of product feature relevance and the implications to product platform and product family design. Specifically, this work aims to:

- Translate a product feature classification from the market-driven domain to the detailed engineering domain.
- Determine the optimal product platform sharing decisions based on the market-driven evolution of product features and customer preferences.

### 6.3 Methodology

The methodology proposed in this work (Fig. 6.1) aims to guide product family design by linking *Temporal Market-Driven Responses* relating to product feature trends with *Engineering Design Optimization* objectives such as product platforming and commonality decisions. As presented in Sect. 6.2.3, there are well-established metrics for evaluating commonality decisions in product family design. However, temporal, market-driven forces are typically not included in these models.

Market-Driven Responses can be a critical design input to product family design by quantifying the evolution of product features in the market space and identifying product features that are *Standard*, *Nonstandard*, or *Obsolete*. In the proposed methodology, the *Product Domain* in the *Market-Driven Response* step refers to the results of a data mining-driven approach to modeling the relevant/irrelevant product features across a wide array of products existing in that domain.

#### 6.3.1 Level 1: Temporal Market-Driven Preferences

Level 1 of the proposed methodology is based on a knowledge discovery in databases (KDD) framework. KDD is the umbrella term used to describe the

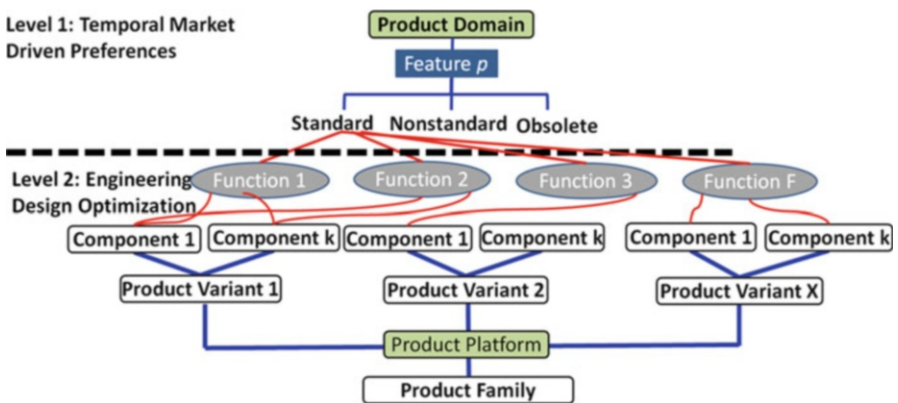


Fig. 6.1 Linking market-driven response with engineering design optimization

Time $t_1$					Time $t_n$				
Feature 1	Feature 2	...	Feature p	Class	Feature 1	Feature 2	...	Feature p	Class

Fig. 6.2 Time series product data containing product features and class

sequential steps of *Data Acquisition* → *Data Selection and Cleaning* → *Data Transformation* → *Data Mining/Pattern Discovery* → finally leading to the *Interpretation* and *Evaluation* of the resulting model. This data-driven approach to modeling will enable designers to understand the temporal changes in the market space relating to product preferences and use this knowledge in the design of next-generation product families. The sequence of the KDD steps will now be expounded upon:

**6.3.1.1 KDD Step 1: Data Acquisition**

The data in the proposed methodology represents structured, time series data that exists within a company’s database or acquired online through publicly available customer product preference websites using automated data acquisition techniques (Tucker and Kim 2011a). The two types of data employed in the proposed methodology are *structured* and *unstructured* data.

*Structured* data typically refers to data that can be conceptualized using an Entity-Relationship structure and easily stored in a Database Management System (Chen 1976). The Entity-Relationship Model is an example of the format of structured data where the *entity* (e.g., product domain) is *related* to certain features (e.g., product features).

Figure 6.2 is an example of time series structured data suitable for the proposed methodology, where each column (1,  $P$ ) at time  $t_i$  is defined by a unique product feature ( $j$ ). The last column containing the *Class* variable represents the dependent/output variable which is influenced by the levels/values of the product features. Examples of a class variable could be a market price segment (>\$199, \$99–\$199, \$0–\$99) or a purchasing decision (purchased, not purchased), etc. The proposed methodology assumes that the product features ( $j$ ) can be categorical or numeric in nature, while the class variable is considered categorical for the subsequent data mining algorithm. In the proposed methodology, *structured data* will be used to

**Fig. 6.3** Unstructured data of a cell phone product review

<p><b>Pros:</b>          Lightweight          Vivrant 4.8 inch display          4G Connectivity</p> <p><b>Cons:</b>          May be oversized for some          SD card does not come with the phone</p>
--

quantify the relevance of product features in the market space over time (Level 1: Temporal Market-Driven Preferences), which will then help guide Product Family decisions in Level 2 (Engineering Design Objective).

*Unstructured data* on the other hand refers to data that is not well suited for DBMS due to a lack of a well-formed entity-relation model (Buneman et al. 1996). Unstructured data primarily includes text data found in documents, web pages, numeric values, etc. An example of unstructured data would be a product review containing both textual and numeric information.

As can be seen in Fig. 6.3, the information contained in textual data does not have a well-defined feature/class relation found in Fig. 6.2, hereby making traditional data mining classification algorithms ill-suited for such data. However, unstructured data contains extremely valuable information regarding the domain of investigation and can be mined to quantify patterns using Natural Language Processing techniques that will be presented in the Data Mining step in the KDD process. In the proposed methodology, Natural Language Processing will be employed to understand the relation between product feature and component function in Level 2 (Engineering Design Objectives).

### 6.3.1.2 KDD Step 2: Data Selection and Cleaning

The second step in the KDD process aims to minimize noise in the data set that may arise due to missing data values, erroneous/ambiguous features, etc. Data selection and cleaning techniques should be employed for each data type used in the proposed methodology. For categorical features/class found in the *structured data* in Level 1, missing/erroneous values can be addressed by either replacing them with global constant values or the most probable values (based on the frequency of occurrence of a particular feature/class value) (Han et al. 2011). For *unstructured data*, used in Level 2 of the methodology, data selection and cleaning techniques may include text grammatical correction processing for nonword error detection, isolated-word error correction, and context-dependent word correction (Kukich 1992). For example, word corrections could be as straightforward as correcting “cel phne → cell phone” or more complex in trying to determine context and correct “real time *whether* updates → real time *weather* updates.”

### 6.3.1.3 KDD Step 3: Data Transformation

Step 3 of the KDD process is where the data is transformed into acceptable forms for the subsequent Data Mining/Pattern Discovery (Step 4) process. For *structured data* for example, *binning* techniques can help smooth data values of a feature by first sorting and placing feature values in predefined bin categories, where each bin category can be represented by the mean of the feature values in the specific bin (Han et al. 2011). For *unstructured data*, data transformation techniques may include *Stemming*; a process that aims to reduce morphological variants of words to their root form so that word variants can be mapped together (Paice 1994). For example, the words *charger* and *charging* both have the same root word *charge*, which could refer to a phone charger in product design. Data transformation techniques will reduce the noise caused by redundant feature values or words in a large data set.

### 6.3.1.4 KDD Step 4: Data Mining/Pattern Discovery

The Data Mining/Pattern Discovery step in the KDD process is where statistical/machine learning algorithms are employed to the transformed data (from Step 3) in order to discover novel, previously unknown knowledge about the domain of interest. The methodology begins with Phase 1, the iterative evaluation of the *relevance* of product features to the final class variable. Phase 2 presents the classification of the product features deemed *irrelevant* by the data mining predictive model that are then classified as *Standard*, *Nonstandard*, or *Obsolete*. The subsequent product family optimization step is guided by the predictive data mining results that are based on the temporal market-driven product preference data.

#### Phase 1: Iterative Feature Evaluation

The feature classification metric is modeled based on a time series decision tree induction algorithm that captures the emerging product feature trends over time (Tucker and Kim 2011b). *Phase 1* in Fig. 6.4 sequentially tests each product feature's entropy (at each iteration of the algorithm) using  $n$  time-stamped data sets. The calculation of the entropy values are used to rank each product feature's *relevance* to the class variable and also used as the test statistic to classify irrelevant product features in phase 2 of the methodology. In this work, the term *relevance* is defined as a product feature's relationship to the class/output variable.

Given  $n$  time intervals,  $t_1$  to  $t_n$ , each time interval  $t_i$  contains a training data set  $T$ . For training data set  $T$  at time  $t_i$ , each of the feature is tested in order to determine that feature's ability to reduce the uncertainty of the class variable (please see Fig. 6.2). There are several metrics proposed in the literature for evaluating a feature's relation to a class variable including the Gini Index, Gain Ratio,

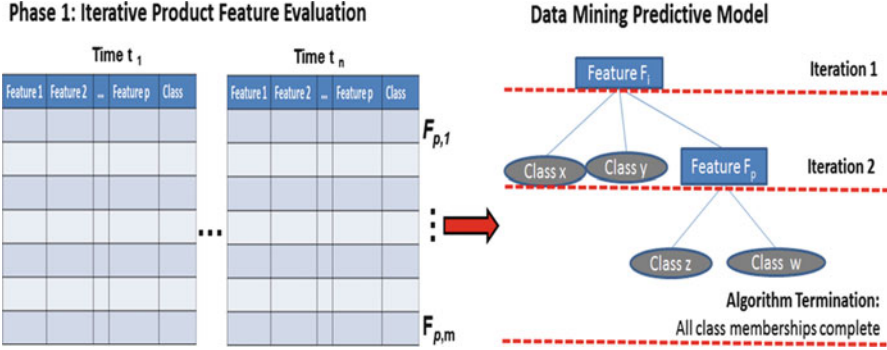


Fig. 6.4 Data Mining model generation based on time series product feature data

Likelihood-Ratio Chi-Squared Statistics, DKM Criterion, Twoing Criterion, etc. (Maimon and Rokach 2005). The methodology proposed in this work employs the *Gain Ratio* metric, although the algorithm is not limited to this metric.

The *Gain Ratio* is a well-established feature evaluation metric for determining the *best split* of the data set at each iteration. The assumption is that both the class variable and product features have values that are mutually exclusive of one another. Also, it is assumed that the variables are categorical or if continuous, can be discretized using existing statistical discretization techniques (Dougherty et al. 1995). The goal of the feature classification algorithm is to iteratively test each product feature for its ability to reduce the uncertainty/randomness of the class variable, generate a decision tree model, and then classify the features that do not show up in the resulting decision tree model as *Standard*, *Nonstandard*, or *Obsolete*.

Given a training data  $T$  set at time  $t_i$ , each with  $n$  features (continuous or discrete) and a class variable  $c_i$ , the *Gain Ratio* is defined as (Quinlan 1992):

$$Gain\ Ratio(X) = \frac{Entropy(T) - Entropy_X(T)}{Split(T)} \tag{6.1}$$

where:

$$Entropy(T) = - \sum_{i=1}^q p(c_i) * \log_2 p(c_i) \tag{6.2}$$

$p(c_i)$  represents the probability (relative frequency) of a class variable  $c_i$  in the training data set  $T$ .

$q$ : represents the number of mutually exclusive class values within the data set.

$$Entropy_X(T) = - \sum_{j=1}^j \frac{T_j}{T} * Entropy(T_j) \tag{6.3}$$

$T_j$ : represents a subset of the training data  $T$  that contains one of the mutually exclusive outcomes of a product feature. For example, if product feature  $X$  is



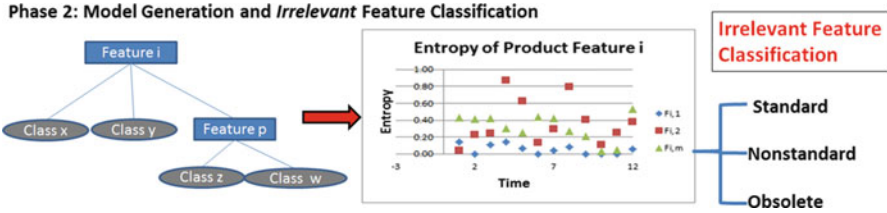


Fig. 6.5 Phase 2: Model generation and irrelevant feature classification

wireless connectivity containing 3 mutually exclusive outcomes (*WiFi, Bluetooth, NFC*), then  $T_j$  represents all the instances in  $T$  that contain one of those outcomes.

$J$ : represents the number of mutually exclusive outcomes for a given feature.

The denominator of the *Gain Ratio* metric,  $Split(T)$  normalizes the numerator, hereby reducing the bias of the metric towards features with a large number of mutually exclusive outcomes ( $T_j$ ).

$$Split(T) = - \sum_{j=1}^j \frac{T_j}{T} * \log_2 \frac{T_j}{T} \tag{6.4}$$

From time periods  $t_1$  to  $t_n$ , the *Gain Ratio* values for each product feature are computed and stored. A time series predictive model is then use to predict which product feature will have the maximum *Gain Ratio* values at future time periods  $t_{n+k}$ , where  $k$  represents the length of time before the next generation of products are to be launched (Tucker and Kim 2011b). Therefore feature  $F_i$ , appearing at the top of the Data Mining Predictive model in Fig. 6.4, represents the product feature with the highest predicted *Gain Ratio*, given a history of stored *Gain Ratio* statistics from time periods  $t_1$  to  $t_n$ . Product feature  $F_p$  at iteration 2 represents the product feature with the highest predicted *Gain Ratio*, given a history stored *Gain Ratio* statistics from time periods  $t_1$  to  $t_n$ . The algorithm continues to partition the original data time series data sets until a homogeneous class distribution exists for each leaf of the Data Mining Predictive Model, as seen in Fig. 6.4. The resulting model is represented as a decision tree, which can be read as a sequence of decision rules by traversing down each unique path of the tree. The resulting model will help design teams determine the specific product feature combinations that yield a particular outcome in the market space (e.g., price).

Phase 2: Model Generation and Irrelevant Feature Classification

Once the Data Mining Predictive Model has been generated from Phase 1, Phase 2 of the methodology (Fig. 6.5) introduces a technique to classify *irrelevant* product features based on the evolution of their importance to future product launches. A challenge in traditional engineering decision support models has been the understanding of the relationship between product features with low model relevance and

the effects on product family design decisions. Phase 2 in Fig. 6.5 overcomes these challenges by utilizing the time history entropy values (calculated and stored at each iteration in Phase 1) to determine the best course of action for *irrelevant* product features. Product feature *irrelevance* is defined simply as product features that do not show up in the resulting predictive model in Fig. 6.5. These product features are classified as either a *Standard Feature*, *Nonstandard Feature*, or an *Obsolete Feature*, with the pseudocode for the algorithm provide below (Tucker and Kim 2011b).

*Start: Iteration  $j = 1$*

1. *If predicted Gain Ratio of Feature  $F_i$  is not the highest, Feature  $F_i$  is considered irrelevant*
2. *Employ Mann–Kendall (MK) trend test for Feature  $F_i$* 
  - a. *If MK  $\tau$  is negative (with  $p$ -value  $<$  alpha), irrelevant classification = Standard*
  - b. *Else If MK  $\tau$  is positive (with  $p$ -value  $<$  alpha), irrelevant classification = Obsolete*
  - c. *Else If MK  $\tau$  is positive/negative (with  $p$ -value  $>$  alpha), irrelevant classification = Nonstandard*
3. *While data set/subset does not contain a homogeneous class*
  - a. *Split the data set into subsets based on the number of mutually exclusive values of the feature with the highest Gain Ratio from Step 2*
  - b.  *$j = j + 1$  and revert to Step 2 for each data subset*
4. *End Tree, Classify Irrelevant Feature  $F_i$  based on highest variable value (SF  $_{t=1, \dots, n}$ ; NF  $_{t=1, \dots, n}$ ; OF  $_{t=1, \dots, n}$ )*

In order to classify product features, the emerging predictive power of each product feature must be quantified over time (i.e., each product feature's relevance to the class variable over time as seen in Fig. 6.5). This is achieved by employing the nonparametric Mann–Kendall trend test, mathematically represented as (Kendall and Gibbons 1990):

$$\tau = \frac{S}{\frac{1}{2}n(n-1)} \quad (6.5)$$

where

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \text{sgn}(x_j - x_i) \quad (6.6)$$

$n$ : represents the total number of time series data points

$x_j$ : represents the data point one time step ahead

$x_i$ : represents the current data point

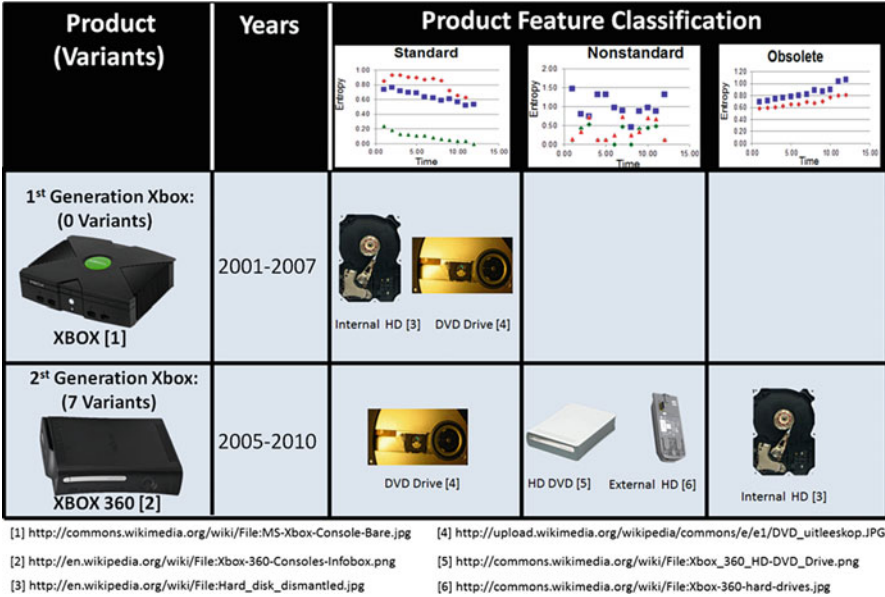


Fig. 6.6 Examples of product feature classification in consumer electronics

$$sgn = \begin{cases} 1 & \text{if } (x_j - x_i) > 0 \\ 0 & \text{if } (x_j - x_i) = 0 \\ -1 & \text{if } (x_j - x_i) < 0 \end{cases} \quad (6.7)$$

The Mann–Kendall begins with a null hypothesis of no trend and rejects or does not reject the null hypothesis based on the resulting p-value and level of significance ( $\alpha$ ).

The product feature classification framework relies on the results of the Mann–Kendal trend test to quantify the magnitude of the relationship between a given product feature and the output (class) variable. The 3 product feature classification categories are provided below with an application example in Fig. 6.6, illustrating the how the product feature classification could be used to guide enterprise level product family design decisions.

*Standard Feature (SF)*

A feature  $F_s$  is defined as standard if it does not show up in the final decision tree model (as seen in Fig. 6.5) and subsequent tests of the times series Entropy statistics using data from  $t_1, \dots, t_n$  (acquired at each iteration of the model generation process) reveal a monotonically decreasing trend. The Mann–Kendall trend detection test is used as the statistical measure to detect trends. If a monotonically decreasing trend is detected by the Mann–Kendall trend test, this means that despite Feature  $F_s$ 's absence from the decision tree model in Fig. 6.5, it is consistently gaining relevance over time and should therefore be considered as a candidate to be included in the

product platform decision in Level 2 of the methodology (Engineering Design Optimization level). The Mann–Kendall would return a negative  $\tau$  and a p-value below the significance level ( $\alpha$ ).

A binary variable ( $SF$ ) is defined for the Standard Feature classification that represents the results of the Mann–Kendall trend test at each iteration  $j$ . That is, if the Mann–Kendall trend test determines that product feature  $F_s$  has a monotonically decreasing entropy trend at iteration  $j$ , the binary variable ( $SF$ ) assumes a value of 1, otherwise 0. Each iteration of the Standard Feature classification  $SF_j$  is weighted based on the number of supporting instances in the data set ( $T_j/T$ )

$$SF(t = 1, \dots, n) = \sum_{j=1}^n SF_j \cdot \left(\frac{T_j}{T}\right) \quad (6.8)$$

A product feature with a *Standard* classification could be considered for the product platform integration during the product family design process. The engineering components providing the functionality for this product feature could be shared across multiple products within the product family. For example, Fig. 6.6 shows 2 product features that were integrated into the 1st-generation Xbox platform: *Internal Hard Drive* and a *DVD player*. Other video game manufacturers such as Sega opted not to include DVD player functionality as a standard product feature in their video game platform (Sega Dreamcast) which contributed to the failure of the system, and ultimately the company as a whole (Aoyama and Izushi 2003). Understanding when to make product features standard (part of the product platform) or nonstandard (modular design that can be replaced/removed) is extremely critical to market success as will be seen in the following classification definitions.

#### *Nonstandard Feature (NF)*

A feature  $F_n$  is defined as Nonstandard if it does not show up in the final decision tree model (as seen in Fig. 6.5) and subsequent tests of the times series Entropy statistics using data from  $t_1, \dots, t_n$  (acquired at each iteration of the model generation process) reveal no discernible trend pattern. The Mann–Kendall trend detection test is used as the statistical measure to detect trends. If no discernible trend is detected by the Mann–Kendall trend test, this means that despite Feature  $F_n$ 's absence from the decision tree model in Fig. 6.5, Feature  $F_n$  exhibits inconsistent relevance patterns through time and should therefore be investigated during the detailed Engineering Design process (Level 2 of the methodology). The Mann–Kendall trend test would return a p-value *above* the significance level ( $\alpha$ ) (which would mean that we do not reject the null hypothesis of no trend). A binary variable ( $NF$ ) is defined for the Nonstandard Feature classification that represents the results of the Mann–Kendall trend test at each iteration  $j$ . That is, if the Mann–Kendall trend test determines that product feature  $F_i$  has no discernible entropy trend at iteration  $j$ , the binary variable ( $NF$ ) assumes a value of 1, otherwise 0. Each iteration of the Nonstandard Feature classification  $NF_j$  is weighted based on the number of supporting instances in the data set ( $T_j/T$ ).

$$NF(t = 1, \dots, n) = \sum_{j=1}^j NF_j \cdot \left(\frac{T_j}{T}\right) \quad (6.9)$$

As opposed to having product variants share the same component addressing a given *Nonstandard* product feature, designers should avoid component sharing decisions within a product family, and instead develop unique components for each product variant in the product family. The engineering components providing the functionality for this product feature could therefore subsequently be replaced, upgraded, or removed altogether if the product feature eventually becomes obsolete in the market space. Figure 6.6 shows 2 product features of the 2nd-generation Xbox (Xbox 360) that had a modular design that was not shared between product variants within a product family: *HD-DVD player* and *Removable External Hard Drive*. During the video game console wars in the mid-2000s, two competing media formats were in direct competition with one another: the *Blu-ray* and *HD-DVD* (Brookey 2007). With the uncertainty of a clear winner in the market space, Microsoft opted for a modular add-on HD-DVD device (see Fig. 6.6) that could seamlessly integrate with the Xbox 360 product variants in the market space if High Definition media consumption was desired by consumers. The add-on HD-DVD was discontinued soon after it was clear that Sony's Blu-ray format had won the next-generation media platform wars (Daidj et al. 2010), making the HD-DVD modular device for the Xbox 360, *Obsolete*. If Microsoft had made the decision early on in the Xbox 360 product design process to integrate the HD-DVD player into the product platform, shared across multiple product variants, an entire redesign of the Xbox 360 system may have resulted after the HD-DVD product feature failed in the market space. The *Removable External Hard Drive* also allowed different variants of the Xbox 360 to have different storage capacities (20 GB, 60 GB, 120 GB, etc.), allowing greater customization in the market space, while keeping the core Xbox 360 relatively unchanged (Microsoft Inc. 2007; Farkas 2009).

### *Obsolete Feature (OF)*

A feature  $F_o$  is defined as obsolete if it does not show up in the final decision tree model (as seen in Fig. 6.5) and subsequent tests of the times series Entropy statistics using data from  $t_1, \dots, t_n$  (acquired at each iteration of the model generation process) reveal a monotonically increasing trend. The Mann–Kendall trend detection test is used as the statistical measure to detect trends. If a monotonically increasing trend is revealed by the Mann–Kendall trend test, this means that despite Feature  $F_o$ 's absence from the decision tree model in Fig. 6.5, it is consistently losing relevance over time and should therefore be investigated during the detailed Engineering Design process in Step 3. The Mann–Kendall trend test would return a positive  $\tau$  and a p-value below the significance level ( $\alpha$ ). A binary variable ( $OF$ ) is defined for the Obsolete Feature classification that represents the value results of the Mann–Kendall trend test at each iteration  $j$ . That is, if the Mann–Kendall trend test determines that product feature  $F_o$  has a monotonically increasing entropy trend

at iteration  $j$ , the binary variable ( $OF$ ) assumes a value of 1, otherwise 0. Each iteration of the Obsolete Feature classification  $OF_j$  is weighted based on the number of supporting instances in the data set ( $T_j/T$ ).

$$OF(t = 1, \dots, n) = \sum_{j=1}^j OF_j \cdot \left(\frac{T_j}{T}\right) \quad (6.10)$$

The final classification of a feature (that does not show up in the decision tree model in Phase 2 of Fig. 6.5) is achieved by summing across all iterations of each of the feature classification variables (Standard, Nonstandard, and Obsolete) and selecting the variable with the highest value.

A product feature with an *Obsolete* classification indicates that it has little market-driven significance over time. The engineering components providing the functionality for this product feature can be considered candidate components to be removed in next-generation product designs. Figure 6.6 shows 1 product feature that was initially part of the 1st-generation Xbox platform, characterized as *Obsolete* in the 2nd-generation Xbox platform. The 2nd-generation Xbox platform (Xbox 360) launched in 2005 without an internal hard drive, making this product feature obsolete to the Xbox 360 platform. Microsoft opted for modular hard drives (External HD in Fig. 6.6) that could be replaced or upgraded at a certain price (Farkas 2009). This enabled Microsoft to discontinue an obsolete component (the internal hard drive), while creating a product family of Xbox 360 s that served different consumer market segments based on the technical capabilities of the Xbox 360 platform [platforms came with no External Hard Drive, 20 GB External Hard Drive and 100 GB Hard Drive (Farkas 2009)].

The methodology proposed in this chapter aims to address the link between the evolution of product feature relevance and the implications to product platform and product family design. Specifically, this work aims to:

- Translate a product feature classification from the market-driven domain to the detailed engineering domain.
- Determine the optimal product platform sharing decisions based on the market-driven evolution of product features and customer preferences.

## 6.3.2 Level 2: Engineering Design Optimization

### 6.3.2.1 Mapping Product Feature Space to Engineering Design Space

From the resulting Data Mining Predictive Model from Level 1 of the methodology, product features will either:

1. Be part of the predictive model and therefore considered relevant product features.

2. Be omitted from the predictive model and be classified as:

- (a) Standard
- (b) Nonstandard
- (c) Obsolete

Level 2 of the product feature classification framework plays a vital role in mapping market-driven, product feature preference trends to engineering design specifications. The ability of a product family to address market-driven demand is highly dependent on the evolution of product feature preferences over time. While component commonality decisions in product family design aim to provide the optimal configuration of product platforms within a product portfolio, mathematical models often omit evolving product feature preferences in the market space, hereby increasing the risk of product failure when launched. Unlike individually designed products, the market failure of a product family (e.g., due to an unwanted product feature) could result in the redesign of an entire product family (as opposed to just a single product) due to the components shared between product variants. In many real life scenarios, the characteristics of the product feature space (as defined by the customer) significantly differs from the technical characteristics of the engineering product design space. For example, a large-scale data set containing product preference data may contain a product feature such as 8 h *battery life*. For the same product feature however, the technical components used to achieve such functionality of a product's feature would be described by more technical terms such as *60 Whr 6-Cell Lithium-Ion Battery* (Dell Inc. 2012).

The aim is to first map the *Standard*, *Nonstandard*, and *Obsolete* product feature classifications to the *Functions* of a component(s) in a product family, as shown in Fig. 6.7. The *Standard* classification includes all features included in the Data Mining Model, in addition to the SF *irrelevant* feature classifications. The *Non-standard* classification includes all features classified as NF, and the *Obsolete* classification includes all features classified as OF. It is important to note that a feature can have one and only one classification.

The textual descriptions of each product feature will then be compared with the textual description of all functions in a product family to determine which components functions are providing the market-driven product preferences. Each component function  $F_i$  existing in a product family is assumed to be defined by a synonym set  $\{s_1, s_2, \dots, s_s\}$  that describes its technical purpose to the product/product family as a whole. A product feature-function matrix is then created as Table 6.1.

Latent Semantic Analysis is employed to make semantic comparison between the vector of terms characterizing a product feature and those of a product function. LSA not only compares the original vector of textual terms but also their *semantic* meaning and makes the assumption that terms with similar meanings will occur close to each other. Therefore, despite the fact that a product feature term “charge” and the product function “battery” are not identically the same, LSA may quantify the related meaning between the two.

Table 6.1 can be represented by one of two vectors

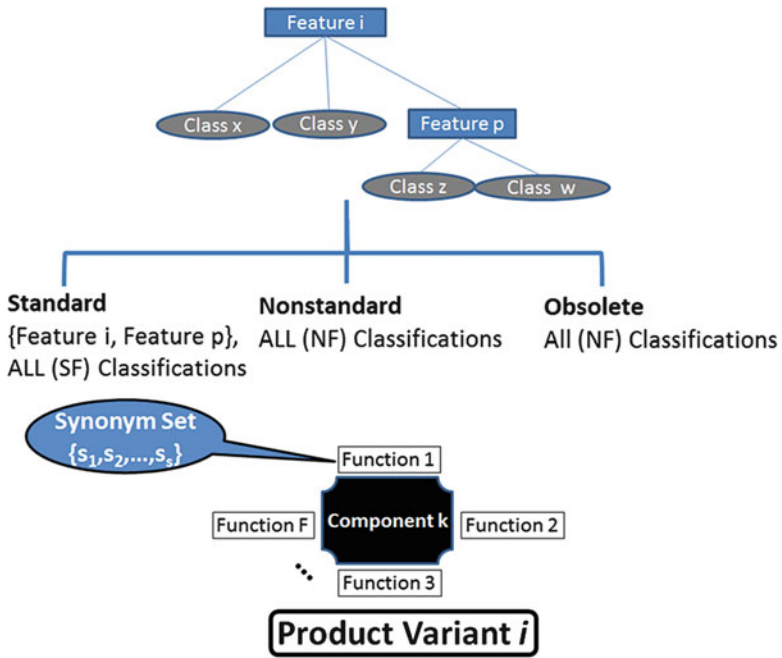


Fig. 6.7 Component function identification

Table 6.1 Product feature-function comparison matrix

	Product feature $P$	Product function 1	...	Product function $F$
Term 1	$C_{1,1}$	$C_{1,2}$	...	$C_{1,n}$
Term 2	$C_{2,1}$	$C_{2,2}$	...	$C_{2,n}$
⋮	⋮	⋮	...	⋮
⋮	⋮	⋮	...	⋮
⋮	⋮	⋮	...	⋮
Term $T$	$C_{m,1}$	$C_{m,2}$	...	$C_{m,n}$

- Semantic term vector (each row of Table 6.1):

$$t_i^T = [c_{i,1} \dots c_{i,n}] \tag{6.11}$$

- Product feature-function Comparison (each column of Table 6.1):

$$f_j = \begin{bmatrix} C_{1,j} \\ \vdots \\ C_{m,j} \end{bmatrix} \tag{6.12}$$



Table 6.1 can be defined as  $X$ , where  $c_{i,j}$  represents the frequency/occurrence of a particular term in the description of either a product feature or product function.

$$X = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{bmatrix} \quad (6.13)$$

The singular value decomposition (SVD) of  $X$  can therefore be represented as (Deerwester et al. 1990):

$$X = T_0 S_0 D'_0 \quad (6.14)$$

where

$X$ : is the term ( $t$ ) by function/feature ( $f$ ) matrix (i.e.,  $X = t \times f$ )

$T_0$ : represents the term ( $t$ ) by rank ( $m$ ) matrix, having orthogonal, unit-length columns ( $T_0' T_0 = I$ )

$S_0$ : is the diagonal matrix of singular values ( $m \times m$ )

$D_0$ : is the rank ( $m$ ) by function ( $f$ ) matrix, having orthogonal, unit-length columns ( $D_0' D_0 = I$ ) (i.e.,  $D_0 = m \times f$ )

$m$ : is the rank of  $X$  ( $\leq \min(t, d)$ )

LSA therefore provides lower-dimension estimates of the original high-dimension space which then enables a comparison of the semantic meaning (beyond just simple term matching) between a product feature and a product's function using similarity metrics such as *cosine similarity* (Tucker and Kang 2012).

Once the market-driven product features have been mapped to specific product functions using SVD, a detailed function-component analysis must be performed. A component-function matrix representation is used to quantify the relationships/interactions between components. A majority of the literature reviewed in this work typically focus on component sharing optimization within and between a family of products. By employing the market-driven product feature classification methodology presented in the previous section, engineering design decisions relating to product family optimization can be guided by emerging product preferences in the market space. The DSM has been employed extensively in the design community to represent interactions among products/processes in a design process (Browning 2001). Examples of interactions captured by the DSM framework include *Spatial* (associations relating to the physical location of elements), *Energy* (energy transfer/exchange between elements), *Information* (data/signal exchanges between elements), and *Material* (material exchange between elements) (Pimpler and Eppinger 1994).

The first step is to determine which functions of a product relate to specific component(s). There may be some components that perform more than one specific function, even across product variants as can be seen in Fig. 6.8. For example, unlike the *60 Whr 6-Cell Lithium-Ion Battery* that supplies electrical energy to a

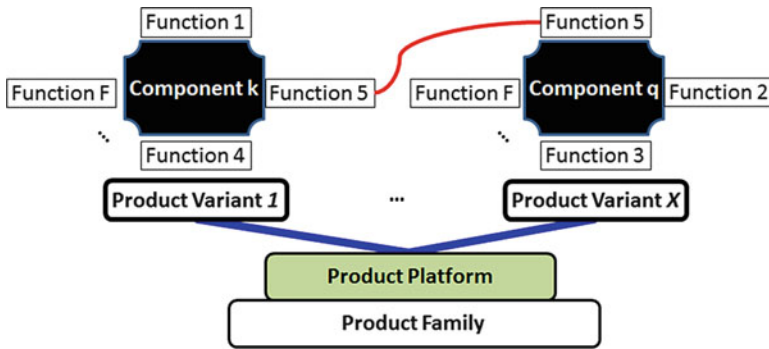


Fig. 6.8 Function-component analysis in a product family

Table 6.2 Component-function interaction matrix

	Function 1	Function 2	...	Function N
Component 1	1			
Component 2				1
...		0		
Component m	0			

product as its *function*, other components such as a DVD super drive component provides multiple functions such as reading media, recording media, and erasing media. Each of these would be considered a unique function of this component. Atomicity is a desired property of the functional decomposition process. It is assumed that designers have a database of components with their individual functions; therefore, the component-function relationship in Table 6.2 will ensure that design teams understand the interactions among components within a product variant and across product variants existing in a product family.

where

1: represents a component *critical* to achieving a specific function

0: represents a component that is *complimentary* to achieving a specific function

The product family optimization problem can be solved using a *quasi-separable*, bi-level optimization model, where the coordination level handles the shared variables (components common to the system) and the product platform level handles the individual product variant optimization problems (each with local objective functions such as cost minimization) (Kim et al. 2003; Tosserams et al. 2006). three binary feature classification variables are included in the objective function to help guide the optimization problem. The *Standard Feature (SF)*, *Nonstandard Feature (NF)*, and *Obsolete Feature (OF)* are modeled as follows.

### Optimization Level 1: Product Family Sharing Level

Minimize

$$\boldsymbol{\varepsilon}_y \quad (6.15)$$

Subject to:

$$g1 : SF_p \cdot \sum_{k \in Q} \left\| \mathbf{y}_p - \mathbf{y}_{p,k}^{Eng} \right\|_2^2 - \boldsymbol{\varepsilon}_y \leq 0 \quad (6.16)$$

Here,

$p$ : product feature from the market-driven Data Mining Predictive Model.

$SF_p$ : binary variable for *Standard Feature* classification (1 if the product feature is deemed *relevant* by the Data Mining Predictive Model and 0 otherwise).

$\mathbf{y}_p$ : linking variable at the product family sharing level that maintains consistency between  $k$  product variant values. The component or variable  $\mathbf{y}_p$  corresponds to function-component map for the specific product feature  $p$ .

$\mathbf{y}_{p,k}^{Eng}$ : the shared variable/component value associated with product variant  $k$  providing product feature  $p$ . This is constant at each iteration in the above formulation that is subsequently updated at the product variant optimization level after each iteration.

$k$ : the  $k$ th candidate product variant that has been identified for component sharing.

$Q$ : the total number of products attempting to share design variables/components  $\mathbf{y}_{p,k}^{Eng}$ .

$\boldsymbol{\varepsilon}_y$ : deviation tolerance between linking variables that is minimized in the objective function.

### Optimization Level 2: Product Variant Optimization Level

Minimize

$$F(x)_{Variant_k} = f_k + \left\| \mathbf{y}_p^U - \mathbf{y}_{p,k} \right\| \quad (6.17)$$

Subject to:

$$\mathbf{g}_k(\mathbf{x}_k, \mathbf{y}_{p,k}) \leq \mathbf{0} \quad (6.18)$$

$$\mathbf{h}_k(\mathbf{x}_k, \mathbf{y}_{p,k}) = \mathbf{0} \quad (6.19)$$

Here,

$p$ : product feature from the market-driven Data Mining Predictive Model.

$NS_p$ : binary variable for *Nonstandard Feature* classification (1 if the product feature is classified as *Nonstandard Feature* by the Data Mining Predictive Model and 0 otherwise).

$OF_p$ : binary variable for *Obsolete Feature* classification (1 if the product feature is classified as an *Obsolete Feature* by the Data Mining Predictive Model and 0 otherwise).

$f_k$ : local product design objective function (s).

$g_k$ : inequality design constraints.

$h_k$ : equality design constraints.

$x_k$ : design variables local to product variant  $k$ .  $x_k$  is a function of the product feature variables  $NS_p$  and  $OF_p$ .  $NS_p$  and  $OF_p$  are presented here in a general form as the mathematical formulation and inclusion in the optimization model will be highly dependent on the structure of the product family model.

$y_p^U$ : linking variable target value cascaded down to the *Level 2* from *Level 1*; a constant value at each iteration that is subsequently updated with each successful iteration.

$y_{p,k}$ : linking variable at *Level 2* that attempts to match the target linking variable value  $y_p^U$  used to achieve the product feature  $p$ .

$k$ : the  $k$ th candidate product variant that has been identified for component sharing.

For each unique product feature that is included in the product family optimization model has to satisfy the equality constraint:  $H1: SF_p + NS_p + OF_p = 1$ , indicating a single state during the product family optimization model (variable/component sharing, module, or exclusion from the optimization model).

## 6.4 Case Study of a Family of Aerodynamic Particle Separators

Particulate Matter (PM)/particle pollution is a complex mixture of very small particles such as acids, organic chemicals, metals, soil, or dust particles (US EPA 2012). Severe health problems can be caused to the heart, lungs, and other organs when PM sizes are 10  $\mu\text{m}$  in diameter or smaller.

Aerodynamic particle separators are devices developed to separate Particulate Matter from the clean air stream, typically by employing centrifugal forces on the particles (Zhang 2005). The case study presented in this methodology is based on an aerodynamic particle separator market segment including applications such as agriculture, industrial, and manufacturing processes. The global market for air cleaning technologies has exceeded \$7 Billion and continues to rise (Parker 2006). The diverse operating conditions and preferences of customers make product standardization a challenge. Customized solutions for aerodynamic particle separators are typically used to solve the wide range of market segments (Fig. 6.9).

This case study aims to investigate the feasibility of employing the proposed Data Mining-Driven product family design methodology to help:

- Quantify the evolution of product feature characteristics over time.
- Develop a Data Mining predictive model of *relevant* product features for future product family designs.



**Fig. 6.9** Aerodynamic particle separator market segments [adapted from (Barker 2008)]

**Table 6.3** Sample data set for aerodynamic particle separator

Product features					Environmental features				Class variable
Q	Delta p <sub>max</sub>	L <sub>max</sub>	AF <sub>max</sub>	N <sub>max</sub>	F(dp)	Rho <sub>p</sub>	T <sub>air</sub>	P <sub>air</sub>	Efficiency
3	2259	0.16	1.2	27	MMD <sub>35</sub> - GSD <sub>1.9</sub>	2181	142	138	85–90 %
1	923	0.02	0.76	27	A1	2977	218	94	85–90 %
1	753	0.53	0.01	37	A4	2022	279	118	85–90 %
3	2082	0.69	0.13	10	Limestone	2133	50	130	85–90 %
4	2890	0.64	1.01	44	MMD <sub>10</sub> - GSD <sub>1.8</sub>	2714	473	199	85–90 %

- Classify product features as *Standard*, *Nonstandard*, or *Obsolete*.
- Investigate how product feature classification influences product family sharing and optimization decisions.

For more details regarding this case study, please see references Barker (2008) and Tucker et al. (2010).

### 6.4.1 Level 1: Temporal Market-Driven Preferences

Table 6.3 above presents a snapshot of the structure of the data set for the aerodynamic particle separator for one instant in time ( $t_i$ ), where:

$Q$ : air flow rate ( $\text{m}^3/\text{s}$ )

$\Delta P_{\text{max}}$ : maximum allowable change in pressure drop/airflow restriction (Pa)

$L_{\text{max}}$ : total allowable length of the system (m)

$AF_{\text{max}}$ : maximum allowable face area perpendicular to air flow direction ( $\text{m}^2$ )

$N_{\text{max}}$ : maximum number of aerodynamic particle separator units in one module (#)

$F(d_p)$ : particle size distribution (%)

$\rho_p$ : particle density ( $\text{kg}/\text{m}^3$ )

$T_{\text{air}}$ : air temperature (°C)

$P_{\text{air}}$ : air pressure (kPa)

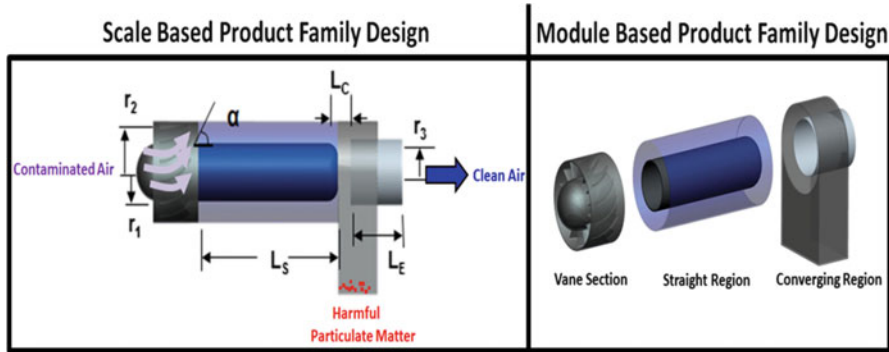


Fig. 6.10 Uniflow aerodynamic particle separator design [Augmented from Tucker et al. (2010)]

The data set contains nine features relating to the aerodynamic particle separator (five of which are related to the physical design of the system while the remaining four are related to the environmental conditions that the system will perform under). The product features will help guide the product family optimization process by suggesting candidate components for sharing or displacement. The Environmental Features will serve as the design constraints of the model. The *class* variable here is *Efficiency* which is defined as the total amount of particulate matter that a system is able to separate from clean air.

The data in Table 6.3 is mined for emerging product feature trends in the market space by quantifying the *relevance* of product features over time. A Data Mining predictive model is generated that helps guide Level 2 of the product family design methodology: Engineering Design Optimization.

### 6.4.2 Level 2: Engineering Design Optimization

The aerodynamic particle separator has a fan system downstream (attached to the radius  $r_3$  in Fig. 6.10) and operates by pulling contaminated air from upstream into the system. The contaminated air (composed of clean air and particulate matter) enters the vane section (Fig. 6.10) causing the particles to rotate based on the angle of the vane section (Barker 2008). The straight section in Fig. 6.10 is designed to increase the separation (centripetal acceleration and inertia) between the particulate matter and the clean air. The clean air particles enter the *converging region* of the particle separator, leaving the particulate matter to collect in the storage bunker in Fig. 6.10.

Figure 6.10 presents 2 approaches to the product family design, *Scale based* and *Module based*. *Scale-based* product family is where a product platform is “stretched” or “shrunk” in one or more dimensions in order to satisfy a market need (Simpson et al. 2006). Design/scaling variables are formulated in the product

family optimization model to achieve such scalability. *Module-based* product family design on the other hand creates product variants by adding, replacing, or substituting one or more functional modules from a product platform. A product architecture is considered *modular* if there is a clearly defined mapping of functional elements to physical structures (1-1 or many-1) (Simpson et al. 2006).

The design objectives of each aerodynamic product variant will be influenced by the market-driven Data Mining model. For scale-based product family design, the efficiency of the system can be influenced by altering (*stretching* or *shrinking*) the design variables that make up the physical system (such as length of straight region, inner and outer radii, etc.)

The optimization approach (module vs. scale based) is left up to the design team based on the technical resources and available mathematical models.

### Optimization Level 1: Product Family Sharing Level

The *Product Family Sharing Level* will coordinate the component sharing among product variants of the aerodynamic particle separator by minimizing the tolerance deviation variable of each shared component.

#### Minimize

$$\epsilon_y \quad (6.20)$$

#### Subject to:

$$g1 : SF_p \cdot \sum_{k \in Q} \left\| \mathbf{y}_p - \mathbf{y}_{p,k}^{Eng} \right\|_2^2 - \epsilon_y \leq 0 \quad (6.21)$$

Here,

$p$ : product feature from the market-driven Data Mining Predictive Model.

$SF_p$ : binary variable for *Standard Feature* classification (1 if the product feature is deemed *relevant* by the Data Mining Predictive Model and 0 otherwise).

$\mathbf{y}_p$ : linking variable at the product family sharing level that maintains consistency between  $k$  product variant values. The component or variable  $\mathbf{y}_p$  corresponds to function-component map for the specific product feature  $p$ .

$\mathbf{y}_{p,k}^{Eng}$ : the shared variable/component value associated with product variant  $k$  providing product feature  $p$ .

$k$ : the  $k$ th candidate aerodynamic particle separator product variant

$Q$ : The total number of products attempting to share design variables/components

$\mathbf{y}_{p,k}^{Eng}$ .

$\epsilon_y$ : deviation tolerance between linking variables that is minimized in the objective function.

### Optimization Level 2: Aerodynamic Particle Separator Variants

The engineering design model for the aerodynamic particle separator can be mathematically represented as:

kth Aerodynamic Particle Separator

**Minimize:**

$$F(\mathbf{x})_{variant(k)} = Cost_k - \xi_k + \left\| \mathbf{y}_p^U - \mathbf{y}_{p,k} \right\| \quad (6.22)$$

where

$$\zeta(\mathbf{x}, d_{pi}) = 1 - \exp\left(\frac{\rho_p d_{pi}^2 C_c Q \tan(\alpha) L_s}{9\eta(r_2^2 - r_1^2)}\right) \cdot \exp\left(\frac{\rho_p d_{pi}^2 C_c (V_t^2 G_t(x) + V_z^2 G_r(x))}{\eta V_z}\right) \quad (6.23)$$

$$\xi_k = \sum_{i=1}^N \zeta(\mathbf{x}, d_{pi}) \cdot F(d_{pi}) \quad (6.24)$$

Here,

$\xi_k$ : efficiency of aerodynamic particle separator variant  $k$

$\mathbf{y}_p^U$ : linking variable target value cascaded down to the *Level 2* from *Level 1*; a constant value at each iteration that is subsequently updated with each successful iteration

$\mathbf{y}_{p,k}$ : linking variable at *Level 2* that attempts to match the target linking variable value  $\mathbf{y}_p^U$  used to achieve the product feature  $p$

$k$ : the  $k$ th candidate product variant that has been identified for component sharing

$C_c$ : Cunningham slip correction factor

$d_{pi}$ : diameter of particle  $i$ ,  $\mu\text{m}$

$F(d_p)$ : particle size distribution

$G_t(x)$ : efficiency model geometric relationship between design variables, tangential acceleration

$G_r(x)$ : efficiency model geometric relationship between design variables, radial acceleration

$\rho_p$ : particle density,  $\text{kg}/\text{m}^3$

$\eta$ : air viscosity,  $\text{Pa}\cdot\text{s}$  or  $\text{kg}\cdot\text{m}/\text{s}$

$Q$ : air flow rate,  $\text{m}^3/\text{s}$

$V_t$ : tangential velocity of particle mixture

$V_z$ : axial velocity of particle mixture

$r_1$ : inner tube radius

$r_2$ : inner tube radius

$\alpha$ : vane discharge angle

$L_s$ : maximum pressure drop

**Subject to:**

$$\mathbf{g}_k(\mathbf{x}_k, \mathbf{y}_{p,k}) \leq \mathbf{0} \quad (6.25)$$

$$\mathbf{h}_k(\mathbf{x}_k, \mathbf{y}_{p,k}) = \mathbf{0} \quad (6.26)$$



## 6.5 Results and Discussion

### 6.5.1 Level 1: Temporal Market-Driven Preferences

Figure 6.11 presents the Data Mining Predictive Model based on the temporal market-driven preferences relating to the aerodynamic particle separator. The results in Fig. 6.11 can be interpreted by traversing down each individual branch in the tree until a class variable (Efficiency) is reached (rectangular box). The ovals in Fig. 6.11 represent the product feature that is deemed *relevant* to predicting the market preferences for aerodynamic particle separator efficiency.

Four unique paths can be attained based on the results in Fig. 6.11:

- $L_{max} > 0.49$  then efficiency  $> 85-90\%$
- $L_{max} < 0.49$  and  $Q > 3$  and  $\Delta_{pmax} > 1560$  then efficiency  $> 95\%$
- $L_{max} < 0.49$  and  $Q > 3$  and  $\Delta_{pmax} < 1560$  then efficiency  $> 90-95\%$
- $L_{max} < 0.49$  and  $Q \leq 3$  then efficiency  $> 85-90\%$

Figure 6.11 also provides designers with the appropriate product feature classification (*Standard, Nonstandard, and Obsolete*) of all product features existing in the market space. The next step is to quantify the relationship between product features and product function so that designers can understand how evolving market-driven preferences guide next-generation product platform and product family design decisions.

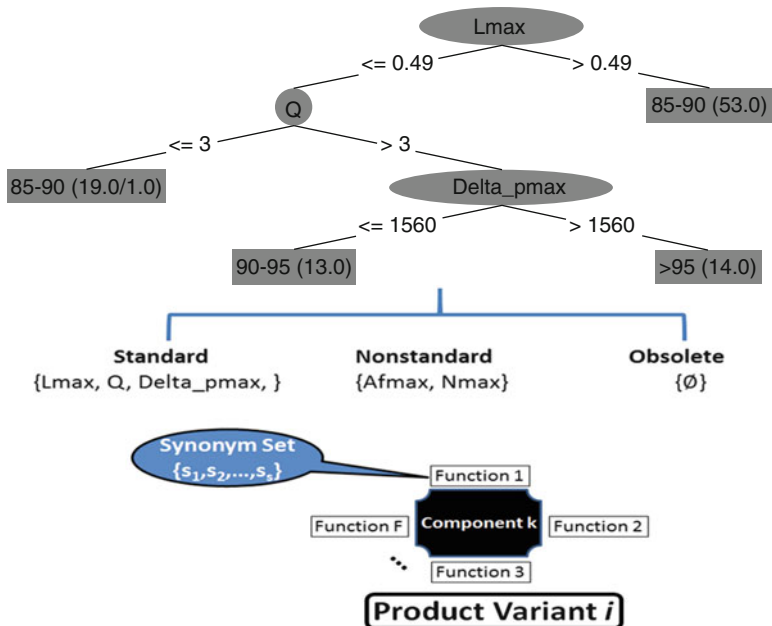


Fig. 6.11 Results from the data mining predictive model [attained using Weka 3.6.6 (Frank et al. 2010)]

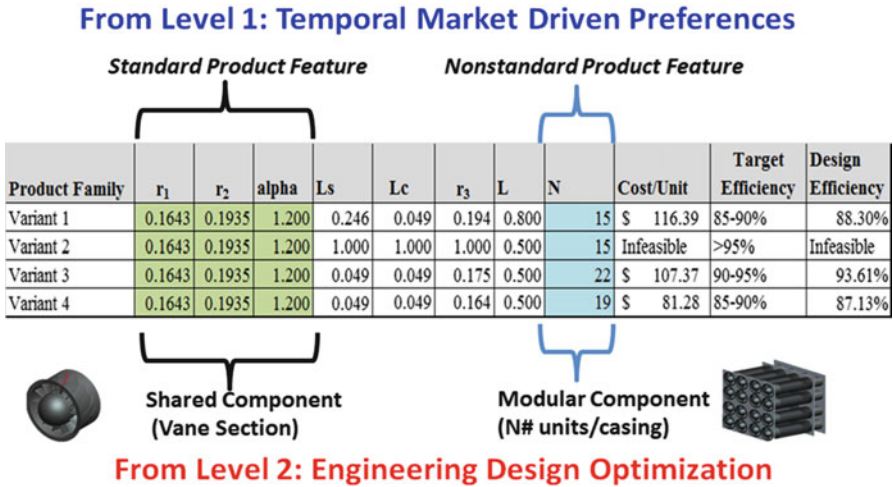
**Table 6.4** Mapping product features of the aerodynamic particle separator to the engineering design space

Product features	Aerodynamic particle separator design space		
	Vane section	Straight region	Converging region
$Q$	0.76	0.71	0.72
$\Delta p_{max}$	0.48	0.5	0.47
$L_{max}$	0.45	0.46	0.43
$Af_{max}$	0.86	0.82	0.78
$N_{max}$	0.49	0.55	0.52
$F(d_p)$	0.35	0.5	0.47
$\rho_p$	0.35	0.5	0.45
$T_{air}$	0.55	0.52	0.57
$P_{air}$	0.63	0.59	0.62

### 6.5.2 Level 2: Engineering Design Optimization

*Mapping Product Feature Space to Engineering Design Space:* Table 6.4 presents the result, employing Latent Semantic Analysis to quantify the relationship between the market-driven product feature space and the product family design space. As described in Sect. 6.2.2, the textual description of each product feature is compared with the functional description of each product module/component providing this function. Table 6.4 represents the *similarity* of a product feature to a component function as measured on a 0–1 scale, where values closer to 1 indicate a stronger relationship to the functionality of the product component/module, while 0 represents a weaker relationship. As can be seen from Table 6.4, the product features are strongly coupled across the entire product architecture. Such insight will help designers understand the market effects of adding, removing, or replacing specific functionality relating to a product. For an *Obsolete* product feature classification, designers would need to ensure that the removal of a particular component/module does not have negative market demand implications. The results in Table 6.4 are consistent with the absence of an *Obsolete* product feature classification from the Data Mining model in Fig. 6.11. The *Standard* and *Nonstandard* product classifications from Fig. 6.11 support the findings from Table 6.4, indicating that all product features are *relevant* to market success at this time. The challenge arises when designers are trying to optimize product family sharing and platforming decisions, which will now be guided by the product feature preference trends in the market space.

Figure 6.12 presents the results from the Data Mining-Driven Product Design methodology. The resulting product feature classifications from Fig. 6.11 helps guide the product family design process by first quantifying the functional relationships between product features and product design variables (Table 6.4) and then suggesting candidate modules/components for sharing decisions. In Fig. 6.12, the vane component is considered a candidate for commonality across product variants due to its relation to the product features deemed relevant by the



**Fig. 6.12** Results from data mining-driven product family design

Data Mining Predictive model in Fig. 6.11. The *Nonstandard* product feature suggests modularity in the design of product variants by including multiple individual products that are housed in a design case. This modularity approach will enable enterprise decision makers to quickly address market needs by increasing/decreasing the number of units housed in a casing in an attempt to meet emerging customer preferences. The *Nonstandard* product feature classification means that there is volatility in the market space, wherein a product may have to undergo modifications (modular or scalable) in the future. Figure 6.12 reveals that of the 4 product segments suggested by the market space, the designers can only satisfy the performance requirements of 3 of those markets, as Product Variant 2's design (Target efficiency > 95 %) is infeasible at the engineering design level. Such design insights enable design teams to focus on developing a product portfolio that both capitalizes on product standardization through *guided* commonality decisions, and at the same time, providing customized *solutions* to the market that meet customer expectations (efficiency requirements).

## 6.6 Conclusions

This chapter introduces a market-driven, product family design framework based on product feature classification framework as it relates to engineering component selection and product family design. Product features are classified as *Standard*, *Nonstandard*, or *Obsolete*, with a given classification having different implications in the product family design process. A component-function matrix is presented to quantify the relationships and interactions among components as it relates to

product specific features. By employing Natural Language Processing techniques, the product feature space can be mapped to the engineering design space for optimal product platform decisions that incorporate market-driven objective. The methodology aims to aid design teams in the efficient modeling of customer preferences and designing of subsequent product portfolios.

## References

- Agard B, Kusiak A (2004) Data-mining-based methodology for the design of product families. *Int J Prod Res* 42:2955–2969
- Alizon F, Shooter S, Simpson T (2009) Assessing and improving commonality and diversity within a product family. *Res Eng Des* 20:241–253. doi:[10.1007/s00163-009-0066-5](https://doi.org/10.1007/s00163-009-0066-5)
- Aoyama Y, Izushi H (2003) Hardware gimmick or cultural innovation? Technological, cultural, and social foundations of the Japanese video game industry. *Res Pol* 32:423–444. doi:[10.1016/S0048-7333\(02\)00016-1](https://doi.org/10.1016/S0048-7333(02)00016-1)
- Barker D (2008) Development of an optimization design platform for aerodynamic particle separators. MS, University of Illinois at Urbana-Champaign
- Boas RC (2008) Commonality in complex product families: implications of divergence and lifecycle offsets. Thesis, Massachusetts Institute of Technology
- Bouchereau V, Rowlands H (2000) Methods and techniques to help quality function deployment (QFD). *Benchmark Int J* 7:8–20. doi:[10.1108/14635770010314891](https://doi.org/10.1108/14635770010314891)
- Braha D (2001) Data mining for design and manufacturing. Kluwer Academic Publishers, The Netherlands, P.O. Box 17, 3300 AA Dordrecht
- Brookey RA (2007) The format wars. *Convergence* 13:199–211. doi:[10.1177/1354856507075245](https://doi.org/10.1177/1354856507075245)
- Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Trans Eng Manag* 48:292–306. doi:[10.1109/17.946528](https://doi.org/10.1109/17.946528)
- Buneman P, Davidson S, Hillebrand G, Suciu D (1996) A query language and optimization techniques for unstructured data. *SIGMOD Rec* 25:505–516. doi: <http://doi.acm.org/10.1145/235968.233368>
- Chen PP (1976) The entity-relationship model – toward a unified view of data. *ACM Trans Database Syst* 1:9–36
- Collier D (1981) The measurement and operating benefits of component part commonality. *Decis Sci* 12(1):85–96
- Daidj N, Grazia C, Hammoudi A (2010) Introduction to the non-cooperative approach to coalition formation: the case of the Blu-Ray/HD-DVD standards' war. *J Media Econ* 23:192–215. doi:[10.1080/08997764.2010.527206](https://doi.org/10.1080/08997764.2010.527206)
- Deerwester S, Dumais ST, Furnas GW et al (1990) Indexing by latent semantic analysis. *J Am Soc Inform Sci* 41:391–407
- Dell Inc. (2012) DELL 60 WHR 6-Cell Lithium-Ion Battery for Dell Inspiron 14z (1470)/15z (1570) Laptops: Laptop Accessories | Dell. <http://http://accessories.us.dell.com/sna/productdetail.aspx?c=us&l=en&s=bsd&cs=04&sku=312-0823>. Accessed 1 Feb 2012
- Dougherty J, Kohavi R, Sahami M (1995) Supervised and unsupervised discretization of continuous features. In: *Proceedings of the 12th international conference on machine learning (ICML-1995)*, San Francisco, CA
- Farkas BG (2009) *The Xbox 360 pocket guide*. Pearson Education, Upper Saddle River, NJ
- Frank E, Hall M, Holmes G et al (2010) Weka-a machine learning workbench for data mining. In: Maimon O, Rokach L (eds) *Data mining and knowledge discovery handbook*. Springer, Berlin, pp 1269–1277
- Han J, Kamber M, Pei J (2011) *Data mining: concepts and techniques*. Elsevier, Amsterdam

- Jiao J, Tseng MM (2000) Understanding product family for mass customization by developing commonality indices. *J Eng Des* 11:225–243
- Kendall M, Gibbons JD (1990) Rank correlation methods, 5th edn. A Charles Griffin Title, London
- Kim HM, Michelena NF, Papalambros PY (2003) Target cascading in optimal system design. *Trans ASME J Mech Des* 125:474–480
- Kota S, Sethuraman K, Miller R (2000) A metric for evaluating design commonality in product families. *ASME J Mech Des* 122(4):403–410
- Kukich K (1992) Techniques for automatically correcting words in text. *ACM Comput Surv* 24:377–439. doi:[10.1145/146370.146380](https://doi.org/10.1145/146370.146380)
- Kwong CK, Bai H (2003) Determining the importance weights for the customer requirements in QFD using a fuzzy AHP with an extent analysis approach. *IIE Trans* 35:619–626. doi:[10.1080/07408170304355](https://doi.org/10.1080/07408170304355)
- Martin M, Ishii K (1996) Design for variety: a methodology for understanding the costs of product proliferation. In: *ASME design engineering technical conferences and computers in engineering conference – design theory and methodology*, ASME, Irvine, CA, 18–22 Aug. Paper no. 96-DETC/DTM-1610
- Martin MV, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. *Res Eng Des* 13(4):213–235
- Maimon O, Rokach L (2005) *Data mining and knowledge discovery handbook*, 1st edn. Springer, Berlin
- Martin MV, Ishii K (1997) Design for variety: development of complexity indices and design charts. *Advances in design automation*, ASME, Paper No. DETC97/DFM-4359
- Meyer MH, Lehnerd AP (1997) *The power of product platforms*. Free Press, Boston
- Microsoft Inc. (2007) Microsoft unveils xbox 360 elite. <http://www.microsoft.com/en-us/news/press/2007/mar07/03-27xbox360elitepr.aspx>. Accessed 21 Jul 2012
- Moon SK, Kumara SRT, Simpson TW (2006) Data mining and fuzzy clustering to support product family design. In: *Proceedings of the ASME design automation conference*
- Paice CD (1994) An evaluation method for stemming algorithms. In: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer, New York, NY, pp 42–50
- Parker PM (2006) *The 2007–2012 world outlook for dust collection and other air purification equipment and parts for cleaning incoming air*. ICON Group International, San Diego, CA
- Pimpler TU, Eppinger SD (1994) Integration analysis of product decompositions. <http://dspace.mit.edu/handle/1721.1/2514>. Accessed 29 Jan 2012
- Pullmana ME, Mooreb WL, Wardellb DG (2002) A comparison of quality function deployment and conjoint analysis in new product design. *J Prod Innov Manag* 19:354–364
- Quinlan JR (1992) *C4.5: programs for machine learning*, 1st edn. Morgan Kaufmann, San Francisco
- Siddique Z, Rosen DW, Wang N (1998) on the applicability of product variety design concepts to automotive platform commonality. In: *Design theory and methodology*, ASME, Atlanta, GA, Sept 1998. Pap. No DETC98DTM-5661
- Simpson T, Maier J, Mistree F (2001) Product platform design: method and application. *Res Eng Des* 13:2–22. doi:[10.1007/s001630100002](https://doi.org/10.1007/s001630100002)
- Simpson TW, Siddique Z, Jiao J (2006) *Product platform and product family design: methods and applications*. Birkhäuser, New York, NY
- Simpson T, Bobuk A, Slingerland L et al (2012) From user requirements to commonality specifications: an integrated approach to product family design. *Res Eng Des* 23:141–153. doi:[10.1007/s00163-011-0119-4](https://doi.org/10.1007/s00163-011-0119-4)
- Thevenot HJ, Simpson TW (2006) Commonality indices for product family design: a detailed comparison. *J Eng Des* 17(2):99–119
- Tosserams S, Etman LFP, Rooda JE (2006) An augmented Lagrangian decomposition method for quasi-separable problems in MDO. *Struct Multidisc Optim* 34(3):211–227

- Tucker C, Kang S (2012) A bisociative design framework for knowledge discovery across seemingly unrelated product domains. In: 2012 ASME international design engineering technical conferences
- Tucker CS, Kim HM (2008) Optimal product portfolio formulation by merging predictive data mining with multilevel optimization. *Trans ASME J Mech Des* 130:991–1000
- Tucker CS, Kim HM (2009) Data-driven decision tree classification for product portfolio design optimization. *J Comput Inform Sci Eng* 9:041004 doi: [10.1115/1.3243634](https://doi.org/10.1115/1.3243634)
- Tucker C, Kim H (2011a) Predicting emerging product design trend by mining publicly available customer review data. In: Proceedings of the 18th international conference on engineering design (ICED11), vol. 6. pp 43–52
- Tucker CS, Kim HM (2011b) Trend mining for predictive product design. *ASME J Mech Des* 133
- Tucker CS, Kim HM, Barker DE, Zhang Y (2010) A ReliefF attribute weighting and X-means clustering methodology for top-down product family optimization. *Eng Optim* 42:593–616
- US EPA O (2012) Particulate matter | air & radiation | US EPA. <http://www.epa.gov/pm/>. Accessed 22 Jul 2012
- Wacker JG, Trelevan M (1986) Component part standardization: an analysis of commonality sources and indices. *J Oper Manag* 6(2):219–244
- Wang J (2008) Encyclopedia of data warehousing and mining, 2nd edn. IGI Global, Hershey, PA
- Weka (2012) Weka 3 – data mining with open source machine learning software in java. <http://www.cs.waikato.ac.nz/ml/weka/>. Accessed 24 Jul 2012
- Zhang Y (2005) Indoor air quality engineering. CRC, Boca Raton, FL

# Chapter 7

## Platform Valuation for Product Family Design

Seung Ki Moon and Timothy W. Simpson

**Abstract** The valuation of a product increases flexibility in decision-making for developing new products or redesigning existing products and affects product life cycles. Strategic adaptability is essential in capitalizing on future investment opportunities and responding properly to market trends in a dynamic environment. To identify the valuation of a platform in a product family, we investigate strategic module-based platform design using market-based decision-making. The objective of this chapter is to propose a financial model to evaluate design valuation for a platform based on market mechanisms in an uncertain market environment. Real options analysis is applied to value options related to introducing new modules as a platform in a product family. In the proposed model, we use design quality that is determined by customers' preferences and performance utilities for products. To demonstrate implementation of the proposed model, we use a case study and numerical analysis involving a family of mobile products.

### 7.1 Introduction

Product family design allows innovative companies to create customized product roadmaps, to manage designers and component partners, and to develop the next generation of products based on platform strategies (Cronin 2010). By sharing and reusing assets such as components, processes, information, and knowledge across a

---

S.K. Moon (✉)  
School of Mechanical and Aerospace Engineering, Nanyang Technological University,  
Singapore 639798, Singapore  
e-mail: [skmoon@ntu.edu.sg](mailto:skmoon@ntu.edu.sg)

T.W. Simpson  
Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA  
Industrial and Manufacturing Engineering, Penn State University, University Park,  
PA 16802, USA

family of products, companies can efficiently develop a set of differentiated economic offerings while also improving the flexibility and responsiveness of product development (Simpson 2004). Product family design is a way to achieve cost-effective mass customization by allowing highly differentiated products to be developed from a common platform while targeting products to distinct market segments (Shooter et al. 2005).

In uncertain market environments, the valuation of a product increases flexibility in decision-making for developing new products or redesigning existing products and affects product life cycles (Bollen 1999). Design has reflected the requirement changes that are caused by customers' preferences, technologies, economic situations, company's strategies, and competitive moves. Strategic adaptability is essential in capitalizing on future investment opportunities and responding properly to market trends in a dynamic environment (Smit and Trigeorgis 2004).

Market-based product design can consider various and dynamic market environments by capturing dynamic factors, such as customer needs and trends, companies' strategies, regulations, resources, and technologies in product design. To identify the valuation of a platform in a product family, we investigate strategic module-based platform design using market-based decision-making. The value of products depends on market segmentation strategies that are identified by information derived from the relationship between customer needs and functional requirements. Real options valuation provides a rigorous analysis that can be applied to develop a financial model for valuing, managing, and optimally exercising options (Longstaff and Schwartz 2001). Real options analysis is a decision-making method to evaluate design strategies that are affected by company's decision, competitors' action, and new technologies (Smit and Trigeorgis 2004).

The objective of this chapter is to propose a financial model to evaluate design valuation for a platform based on market mechanisms in an uncertain market environment. The proposed model is to facilitate product family design strategies that will maximize the expected profit under uncertain constraints, such as demand, customers' preferences, and regulations. Real options analysis is applied to value modules related to a platform in a product family. In the proposed model, we use design quality that is determined by customers' preferences and performance utilities for products.

The remainder of this chapter is organized as follows. Section 7.2 reviews related literature and background for product family design and market-based design approaches. Section 7.3 describes the proposed financial model to evaluate product family design. Section 7.4 gives a case study and numerical analysis for design valuation involving a family of mobile products. Closing remarks and future work are presented in Sect. 7.5.



## 7.2 Literature Review and Background

### 7.2.1 *Product Family Design*

A product family is a group of related products based on a product platform, facilitating mass customization by providing a variety of products for different market segments cost-effectively (Simpson et al. 2005). A successful product family depends on how well the trade-offs between the economic benefits and performance losses incurred from having a platform are managed. A well-defined platform reduces production costs by improving economies of scale and reducing the number of different components that are used (Simpson et al. 2005; Moon et al. 2008).

Simpson et al. (2001) introduced a method to optimize a platform by minimizing performance loss and maximizing commonality based on a scale-based product family design approach. Johannesson and Claesson (2005) described a configurable product platform design process and model using an operative product structure and a hierarchical function-mean tree to capture parameters describing design information such as rules, variants, requirements, and product configuration possibilities. Thevenot et al. (2007) developed the design of commonality and diversity method (DCDM) to provide designers with recommendations for both the functional and component levels by the inherent trade-off between commonality and diversity during product family and platform development. Moon et al. (2008) introduced a market-based negotiation mechanism to support product family design by determining an appropriate platform level that represents the number of common modules using a dynamic multi-agent system in an electronic market environment. Zacharias and Yassine (2008) proposed a mathematical model for developing and evaluating modular product families to provide maximum market coverage by integrating a conceptual design approach, a product development cost model, an economic model. Moon and McAdams (2009) introduced a method for developing a universal product family through a game theoretic approach in a dynamic market environment by extending concepts from product family design to universal design. Johnson and Kirchain (2011) used a generative cost model to investigate development lead time and costs that can have significant effects on technology choice and lead to substantial cost savings in product families. Rojas Arciniegas and Kim (2012) developed a methodology to identify a set of components containing sensitive information related to security concerns using clustering optimization, while considering component sharing and optimal architecture for a family of products.

### 7.2.2 *Market-Based Design Approaches*

In engineering design and product development, market-based design approaches can provide the ability of investigating additional flexibility and strategic value.

Game theoretic approaches have been applied to model strategic relationships between designers for sharing design knowledge and solving design problems. Real option analysis has offered a natural framework to evaluate the valuation of product design by utilizing managerial flexibility in the valuation process (Gamba and Fusari 2009; Brach 2003).

Xiao et al. (2002) applied game theoretic approaches and design capability indices to model the relationships between engineering teams that were described as cooperative, noncooperative, and leader/follower protocols and facilitate collaborative decision-making during a product realization process. Fernandez et al. (2005) proposed a framework for establishing and managing collaborative design spaces by combining elements of cooperative and noncooperative behavior and formulating strategic and extensive games with utility theory. Kopin and Wilbur (2005) introduced a Bayesian game to model cost sharing in uncertain and incomplete information that were related to producer and consumer attributes such as nature, production costs, players and information, and preferences. Jiao et al. (2006) identified four types of real options based on European options for product family design and developed a valuation framework to evaluate the options of configuration inherent in design using financial analysis. Ford and Sobek (2005) applied real options concepts to product development processes for managing uncertainty through flexibility impacts project behavior, performance, and value. Gamba and Fusari (2009) described a stochastic dynamic framework for valuing the contribution of modularization process and modular operations in the design of systems using real options. Kumar et al. (2009) proposed a market-driven product family design methodology to determine an optimal product offering and platform level strategy based on the estimated demand model in market segmentation grids by evaluating the impact of the variability of products and development cost. Shiau and Michalek (2009) introduced an approach based on Stackelberg game to solve a product design optimization problem for profit maximization under short-run price competition. The proposed approach considered Nash and Stackelberg conditions as design constraints to reflect competitor pricing reactions. Moon et al. (2011) developed a method for designing customized families of services using game theory to model module sharing and decide strategic solutions for selecting modules in dynamic market environments. A coalitional game was employed to model potential module sharing and determine which modules used in the platform provide the most benefit. Jiao (2012) utilized a hybrid real options valuation approach to evaluate the flexibility of product platforms for improving platform planning and investment by integrating product-related options and project-related options under endogenous and exogenous uncertainties. In the next section, the proposed financial model for evaluating the valuation of platform design is discussed in detail.

### 7.3 A Financial Model for Platform Design Valuation

In this chapter, we propose a financial model to investigate design strategies based on the value of the platform design using real option analysis.

#### 7.3.1 Product Family Architecture

The basic idea of modular design is to organize products as a set of distinct modules that can be designed independently and develop a variety of products through the combination and standardization of modules (Ramesh et al. 2002). We assume that a product can be decomposed into modules that provide specific functions, and functions are achieved by the combination of the modules' design variables.

Suppose that a product family consists of  $l$  products,  $F = (P_1, P_2, \dots, P_l)$  and a product,  $i$ , consists of  $m_i$  modules,  $P_i = (x_{i,1}, x_{i,2}, \dots, x_{i,m_i})$ , where  $x_{i,j}$  is a module  $j$  in product  $i$  and consists of a vector of length  $n_m$ ,  $x_{i,j} = (x_{i,j,1}, x_{i,j,2}, \dots, x_{i,j,n_m})$ . The individual scalar components  $x_{i,j,k}$  ( $k = 1, 2, \dots, n_m$ ) of a module  $x_{i,j}$  are called design variables. Each module can be achieved by alternative instances. Let  $b_j$  be an instance of module  $j$  ( $b_j = 1, 2, \dots, B$ ) and  $\mathbf{M}$  be a module instance matrix for the instances of modules in a product family. By introducing a module instance matrix  $\mathbf{M}$ , the product family is represented as

$$\mathbf{PF} = \mathbf{MX} \quad (7.1)$$

where  $\mathbf{M}$  is defined as

$$\mathbf{M}_{i,j} = \begin{cases} b_j & \text{if module } j \text{ is used in a product } i \\ 0 & \text{otherwise} \end{cases} \quad (7.2)$$

In the module instance matrix, if modules are designed by the same instance (i.e., common modules), then the number of  $b_j$  is the same. And, the large number of  $b_j$  indicates the number of alternatives in module  $j$ . Based on the proposed product family architecture, a module instance matrix can be represented as

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \quad (7.3)$$

where  $\mathbf{M}_{11}$  and  $\mathbf{M}_{21}$  are matrixes for common modules, because the common modules should be included in both products.  $\mathbf{M}_{12}$  and  $\mathbf{M}_{22}$  can be a matrix for unique or variant modules in a product family.

### 7.3.2 Company's Profit Model and Platform Strategy Cost

We use sales profits to evaluate company's profit. We assume that the price of a product is determined by the company based on product quality. The product quality can be represented as functions desired by customers. Then, the profit of product  $i$ ,  $\pi_i$ , can be formulated based on sales price, product cost, and demand as follows:

$$\pi_i = (S_i - C_i)D_i \quad (7.4)$$

where  $S_i$  is the sales price of product  $i$ ,  $C_i$  is the product cost of product  $i$ , and  $D_i$  is the sales quantity of product  $i$ . In product family design, product cost depends on platform strategies and design quality. Generally, product cost can be determined by total expected product volume, material cost, direct labor, production resource usage, tooling and capitalization costs, system cost (overhead or indirect costs), and development costs (Magrab 1997). Based on the proposed product architecture as mentioned in Sect. 7.3.1, product cost for product  $i$  is represented by

$$C_i = \mathbf{M}_i \mathbf{L}_i' \mathbf{X}_i' \quad (7.5)$$

where  $\mathbf{M}_i$  is a module instance matrix for product  $i$ ,  $\mathbf{L}_i$  is a vector of module costs for product  $i$ , and  $\mathbf{X}_i$  is module design variables in product  $i$ . The module instance matrix is generated by a feasible set of products and includes a platform strategy to satisfy product requirements in a product family. In product family level, product cost using a platform strategy,  $s_y$ , can be represented as

$$C(s_y) = \mathbf{M}(s_y) \mathbf{L} \mathbf{X} \quad (7.6)$$

where  $\mathbf{M}(s_y)$  is a module instance matrix when  $s_y$  is used for product family design. The different platform strategies are constructed by combining the different modules into common and variant modules.

To develop platform strategies based on common modules, we introduce an expected strategy cost that represents additional costs for developing a new platform for a product family. Such costs could come from redesigning components, creating convenient interfaces, or having some components essentially overdesigned for the most of the product family such that it works sufficiently for one specific product. Let  $A$  be a set of strategies for increasing the platform level, and let  $c_p(s_y)$  be the expected platform strategy cost for strategy  $s_y$  ( $y = 1, 2, \dots, A$ ). Then, the expected platform strategy cost can be calculated as follows (Moon et al. 2008):

$$c_p(s_y) = \eta \times \frac{\sum_{i \in I} C_i^a}{f(I) \times r} \quad (7.7)$$

where  $C_i^d$  is the additional design cost of product  $i$  associated with the new platform,  $\eta$  is a factor for overhead cost, and  $f(I)$  is a strategy weight function as follows:

$$f(I) = \begin{cases} 1, & \text{if a module is unique} \\ I, & \text{otherwise} \end{cases} \quad (7.8)$$

and  $r$  is a volume penalty factor related to product sales quantity. Hence, the expected total product family cost,  $EC$ , for the product family using platform strategy,  $s_y$ , can be calculated by

$$EC(s_y) = \sum_{i \in I} C_i + c_p(s_y) \quad (7.9)$$

where  $C_i$  is the product cost of product  $i$ . For a given set of products, the value of  $c_p(s_y)$  varies depending on the strategy for platform design. The expected platform strategy cost function will be used to determine a platform for a product family and can be developed by various cost functions based on products' characteristics and/or company's strategy in product family development. The next section introduces a financial model for evaluating platform design valuation using real options.

### 7.3.3 A Financial Model

We propose a financial model to evaluate the valuation of product family design using real options analysis in an uncertain market environment. A company tries to maximize profit by identifying module valuation when new platform design for a product family will be introduced into markets. In the proposed financial model, demand can be represented as the source of uncertainty and volatility in a market. We assume that the demand follows a Geometric Wiener Process and has drift,  $\mu$ , for the demand changing (Kamrad and Ritchken 1991). The drift is defined as

$$\mu = r - \frac{\sigma^2}{2} \quad (7.10)$$

where  $r$  is the riskless rate and  $\sigma$  is the instantaneous volatility. Let  $u$  be the rate of moving up,  $d$  be the rate of moving down, and  $ud = 1$ . The demand can move up, move down, or state constant at time  $t$ . The probabilities of movements for the demand at time  $t$  can be obtained as follows (Kamrad and Ritchken 1991):

$$p_1 = \frac{1}{2\lambda^2} + \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma} \quad (7.11)$$

$$p_2 = 1 - \frac{1}{\lambda^2} \quad (7.12)$$

$$p_3 = \frac{1}{2\lambda^2} - \frac{\mu\sqrt{\Delta t}}{2\lambda\sigma} \quad (7.13)$$

where  $\Delta t$  is the length of each time interval and  $\lambda \geq 1$ . If  $\lambda = 1$ ,  $p_2 = 0$ , and the movements of the demand become a binominal model (Cox et al. 1979).

We consider a profit model for a single product. We assume that the demand,  $D_t$ , during time interval,  $t$ , is a variable in company's profit function. Let  $S$  be the sales price and  $C$  be the production cost. We assume that the sales price and the production cost are determined by the company. The production cost is represented as raw materials, labors, logistics, assemblies, financial issues, and regulation. Profit,  $V_{sp}$ , for a product in time interval  $t$  can be defined as

$$V_{sp}(t) = (S - C)D(t) \quad (7.14)$$

Otherwise, we consider a platform that is applied to a product. Let  $d_i$  be the rate of changing demand related to design quality for the product and  $v$  be the rate of variable production cost savings if the platform is applied to a product family. The demand of the product is affected by design quality related to customers' preferences. Let  $A_p$  be the additional cost of introducing a platform per time interval. The additional cost is represented as the research and implementation costs of the new design. The profit per time interval  $t$  for the family product can be calculated as

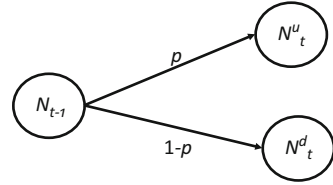
$$V_{fp}(t) = S(1 + d_i)D(t) - (1 - v)(1 + d_i)CD(t) - A_p \quad (7.15)$$

Then, the net benefit from the platform design,  $N_t$ , for introducing a platform during time interval  $t$  can be represented as the maximum of the difference between 0 and two profit Eqs. (7.13) and (7.14):

$$N_t = \max(0, d_iSD_t + (v - d_i + vd_i)CD_t - A_p) \quad (7.16)$$

Interpreting Eq. (7.15), positive values of the net benefit represent an advantage of the family design over single product design. If the net benefit is zero, the company selects a single product to maximize the profit. The net benefit is affected by the volatility rate, the changing demand rate, the saving cost of family design, and the additional cost. To evaluate the valuation of platform design with respect to customers' preferences and family design, sensitive analysis will be performed. In this research, we use a lattice approach to solve the problem (Brach 2003). When  $\lambda = 1$ , the valuation of a platform can be calculated by the value of call option based on the net benefit as follows:

**Fig. 7.1** Call option valuation in the binomial tree



$$F(t, N_t) = \frac{pN_t^u + (1-p)N_t^d}{(1+r_f^t)^{\Delta t}} \quad (7.17)$$

Where  $p$  is a risk natural probability and  $r_f^t$  is a risk-free interest rate at time  $t$ ,  $N_t^u$  is the net benefit of achieving the best case scenario with probability  $p$  at time  $t$ , and  $N_t^d$  is the net benefit of achieving the worst case scenario with probability  $(1-p)$  at time  $t$ . Figure 7.1 shows the binomial tree for assessing the call option.

### 7.3.4 Design Quality

In product design, customers' preference may vary based on specific functional requirements. Functional preference information can help develop market segmentation for a product family by identifying an initial platform based on core common functions. The division of a market into homogenous groups of consumers' preference is known as market segmentation (Meyer and Lehnerd 1997). Because a market segment provides guidelines for determining and directing customer requirements, it can be used to identify the criteria for designing a product family more accurately (Longstaff and Schwartz 2001). The basic development strategy within any product family is to leverage the product platform across products that target multiple market segments. In the initial phase, customers are classified into groups based on their characteristics and preferences. Products are also clustered as groups based on potential suitability for the customers. To evaluate and measure performance of a product, we propose a quality metric that is positively related to product quality, customer preference, and price. In this chapter, we introduce two quality levels to determine the performance of a product (1) marginal quality and (2) full quality. The marginal quality is defined as the level of quality that satisfies minimum functional requirements for customers to perform a job through a product. Customers have zero preference if the service quality of the product is below the marginal quality. The full quality is represented as the level of quality that satisfies maximum functional requirements for customers to pay the price for purchasing a product. The marginal and full qualities are determined by functions depending on customers' preferences in market segments. Figure 7.2 shows two design quality functions of a product for different customers' groups. In between marginal and full qualities, customers have various preferences related to product's quality.

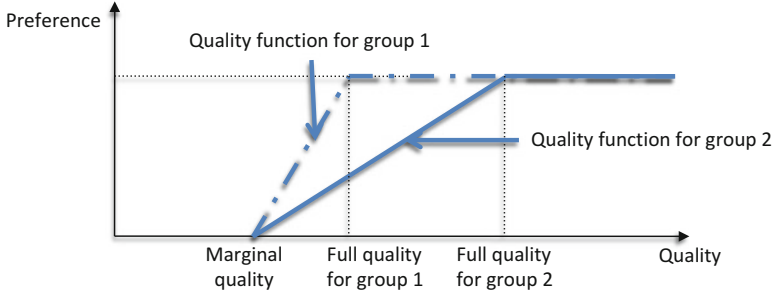


Fig. 7.2 Relationship between preference and quality for a product

We assume that the quality of a product is represented by customers’ preference. To determine the value of customers’ reference related to the product quality,  $Q_p$ , we assume that customers in the market are categorized into two homogenous customers, normal and specific groups. The value of the preference,  $U(Q_p)$ , can be represented by a utility function as follows:

$$U(Q_p) = \begin{cases} 0, & \text{if } Q_p \leq Q_M \\ \frac{f_{n,q}(Q_p) + f_{s,q}(Q_s)}{2}, & \text{if } Q_M < Q_p \leq Q_F^N \\ \frac{1 + f_{s,q}(Q_p)}{2}, & \text{if } Q_F^N < Q_p \leq Q_F^S \\ 1, & \text{if } Q_F^S < Q_p \end{cases} \quad (7.18)$$

where  $Q_M$  is the marginal quality of a product,  $Q_F^N$  is the full quality of a product for a normal customer group,  $Q_F^S$  is the full quality of a product for a specific customer group,  $f_{n,q}$  is a normal quality function, and  $f_{s,q}$  is a specific quality function. The specific quality represents the interaction of product functions: it is a measure that indicates what requirements are needed to make product functions for the specific customer group. In this chapter, the design quality allows us to explore how a particular product platform can best be used to develop a family that provides high qualities to customer groups through the estimation of demand.

In general, market demands can be affected by the quality and price of a product (Krishnan and Zhu 2006). To determine the expected demands for introducing new design at a specific time, we can use the expected preference value and demographics (potential customers) in market segmentation grids that are covered by a product (Moon and McAdams 2009). Based on the expected demand, we can estimate the expected increasing demand rate of a specific product for applying to the proposed financial model. In the next section, the proposed financial model is applied to determine the valuation of a platform strategy using a case study involving a family of mobile products.



### 7.4 Case Study

To demonstrate and validate the proposed model, a family of mobile products consisting of N73, N76, N78-1, and N79-1 is investigated from the Nokia N70 phone family as shown in Fig. 7.3. The Nokia N70 series family products provide a good example of common and variant functions for vision accessibilities as shown in Table 7.1. These products offer the opportunity to create a product family with the vision features as common functions that constitute the product platform. Since the accessible features of the mobile product family are considered as functions for modules or components, the products can be applied to case studies related to universal product family design (Moon and McAdams 2009).

The objective in this case study is to determine the valuation of a new platform strategy in uncertain market environments. The platform design strategy is represented by accessible modules to support persons with vision limitations due to ageing and disabilities. This case study focuses on introducing a new product platform through the addition of accessible modules. Benefit of the proposed product platform is based on the maximum valuation of the proposed additional modules. We also perform sensitivity analysis for the valuation of the platform design strategy with respect to different parameters that reflect the proposed financial model.

Fig. 7.3 Nokia N70 series products (Nokia 2008)



Table 7.1 Vision accessible features for four products (<http://www.nokiaaccessibility.com>)

Vision features		N73	N76	N78-1	N79-1
F1	Tactile key markers	Yes	No	Yes	Yes
F2	Standard key layouts	Yes	Yes	Yes	Yes
F3	Key feedback—tactile	Yes	Yes	Yes	Yes
F4	Key feedback—audible	Yes	Yes	Yes	Yes
F5	Audible identification of keys—when pressed	No	No	No	No
F6	Audible identification of keys—feedback	Yes	Yes	Yes	Yes
F7	Adjustable font style	No	Yes	Yes	No
F8	Adjustable character size	No	Yes	Yes	Yes
F9	Display characteristics (color display)	Yes	Yes	Yes	Yes



Fig. 7.4 Market segmentation grids for the four products

### 7.4.1 Market Analysis and Platform Strategy

Figure 7.4 shows current market segmentation grids for the mobile products with respect to vision features and market prices. The products have different vision accessibility features and market prices depending on market segments. For example, N73 covers no vision impairment and low price market. In Table 7.2, we can consider F2, F3, F4, F6, and F9 as common modules for the phone family. And, F1, F7, and F8 are considered as variant modules.

In this chapter, the company wants to maximize profits by introducing a new platform as accessible modules for the family of mobile products. We facilitate function configuration for developing platform design strategies by identifying relationships between functions and market segments at a conceptual design phase. Using Feature and Component Matrix, we can determine the relationship between vision features and components as shown in Table 7.2. We consider that a cell phone consists of 11 components (Holttä-Otto and De Weck 2007). Among the components, we assume that a main board includes a program for supporting all features.

To develop a new platform consisting of common modules and variant modules, we need to determine valuation of the variant modules (F1, F7, and F8). The valuation of modules can help decide which modules are included to a new platform for increasing benefits and accessible features in product family design. Table 7.3 shows configuration strategies that consist of the variant modules for the phone product family. To determine the expected strategy cost as mentioned in Sect. 7.3.2, we considered the number of components that are related to vision features and use a unit additional cost,  $C^a$ , for each component. For example, since the tactile key marker is related to two components, upper case and keypad, the additional cost of the tactile key marker is  $2 C^a$ . We assume that a factor of overhead cost and a

**Table 7.2** Feature and component matrix for the products

Vision features	Power converter	Power cable	Upper case	Lower case	Speaker	Display unit	Keypad	Microphone	Antenna	Main board	Battery	Component #
F1			×				×					2
F2			×				×			×		3
F3			×				×			×		3
F4			×		×		×			×		4
F5			×		×		×			×		4
F6			×		×		×			×		4
F7			×			×	×			×		4
F8			×			×	×			×		4
F9			×			×	×			×		4

**Table 7.3** The expected additional strategy cost for the platform strategies

Strategy	Additional component design cost					Expected additional strategy cost
	N73	N76	N78-1	N79-1	Total	
S1-F1F7	4C <sup>a</sup>	2C <sup>a</sup>	–	4C <sup>a</sup>	10C <sup>a</sup>	5C <sup>a</sup>
S2-F1F8	4C <sup>a</sup>	2C <sup>a</sup>	–	–	6C <sup>a</sup>	3C <sup>a</sup>
S3-F7F8	8C <sup>a</sup>	–	–	4C <sup>a</sup>	12C <sup>a</sup>	6C <sup>a</sup>
S4-F1F7F8	8C <sup>a</sup>	2C <sup>a</sup>	–	4C <sup>a</sup>	14C <sup>a</sup>	7C <sup>a</sup>

**Table 7.4** Comparison to current market segments and the expected market segments

Platform strategy	N73	N76	N78-1	N79-1
Current	No	Mild	Moderate	Mild
S1	No, mild	Mild	Moderate	Mild, moderate
S2	No, mild	Mild	Moderate	Mild
S3	No, mild, moderate	Mild	Moderate	Mild, moderate
S4	No, mild, moderate	Mild, moderate	Moderate	Mild, moderate

volume penalty factor are 2 and 1, respectively. The expected strategy cost for the product family can be calculated by Eq. (7.7). Table 7.3 shows the results of the expected strategy cost for the platform strategies.

In the vision impairment point of view, Table 7.4 shows a comparison of current market segments and the expected market segments for new platform design strategies. For example, if N73 includes additional features, F1 and F7, as a platform strategy (S1), its expected market segments will cover no and mild segments for the vision impairment.

## 7.4.2 Identify Design Quality

Based on the platform design strategies, the expected design qualities for the products can be calculated by the value of preference as mentioned in Sect. 7.3.3. We assume that the design quality of a product is depended on the number of vision features in the product. We consider customers with vision impairment as the specific group. Figure 7.5 shows the functions of design quality for two customer groups. The marginal quality was determined by the number of common vision features. The full quality of the normal group was determined by the design quality of N73, while the full quality of the specific group was the maximum number of vision features.

Table 7.5 shows the expected design qualities of the products with respect to vision features. The expected preference values of the products for the platform strategies are calculated by Eq. (7.18). For example, the expected design quality

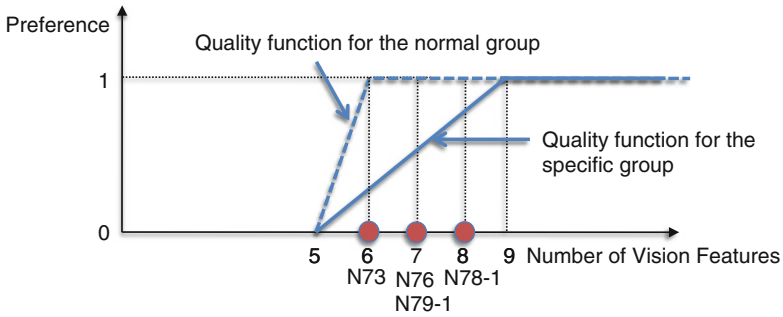


Fig. 7.5 Preference and design quality for the products

Table 7.5 The expected design qualities of the products

Strategy	N73	N76	N78-1	N79-1
Current	0.625	0.75	0.875	0.75
S1	0.75	0.875	0.875	0.875
S2	0.75	0.875	0.875	0.75
S3	0.875	0.75	0.875	0.875
S4	0.875	0.875	0.875	0.875

of N73 in S1 is 0.75, because the number of vision features for N73 is 7 (F1, F2, F3, F4, F6, F7, and F9) and the expected preference value of 7 vision features is 0.75 (0.5 for the specific group and 1 for the normal group). Therefore, we can expect that demand for the mobile products with accessible features will be depended on the number of persons with vision limitations due to age and disabilities.

### 7.4.3 Numerical Analysis

To evaluate the valuation of the vision features, S4 was selected for applying to numerical analysis based on the proposed financial model. We assume that the expected demands for the products are determined by the result of market analysis and the amount of total demands for the cell phone products is 400,000. We assume that the total time horizon for the problem is 10 years and the time interval is 1 year. According to the market analysis in Sects. 7.4.1 and 7.4.2, we assume that the expected demand of the mobile product with accessible features is increased. Suppose that the problem parameters at the current time ( $t = 0$ ) in the case study are as follows:

$S$	=\$400 (average sales price for mobile products)
$C$	=\$320 (average production cost for mobile products)
$D_0$	=400,000 (demand at $t = 0$ )
$A_p$	=\$7 per a product (additional cost when $C^a$ is \$1)
$r$	=5 % (riskless rate)
$\sigma$	=10 % (volatility)
$u$	=1.0226 (the rate of move up)
$d$	=0.9729 (the rate of move down)
$d_i$	=3 % (the rate of the expected increasing demand rate)
$v$	=2 % (the rate of cost saving for product family)
$r_f^t$	=5 % (the rate of a risk-free interest at time $t$ )

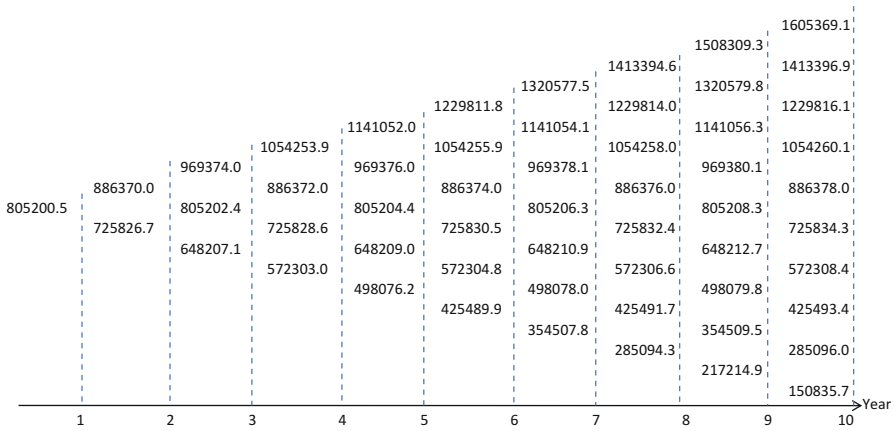


Fig. 7.6 Option value for additional accessible modules in the binomial lattice

We assume that  $\lambda = 1$ . Then, the probabilities of movements for the demand can be calculated by Eqs. (7.10) and (7.12). Therefore,  $p_1 = 0.5712$  (move up) and  $p_3 = 0.4288$  (move down). Using a lattice approach, the valuation of real option for vision accessible modules is estimated to be \$805200.5. This value is represented as the expected worth of introducing additional accessible modules as a platform for the family of the mobile phones for 10 years. A binomial lattice with 10 time steps and 66 nodes is generated to estimate the valuation of the module as shown in Fig. 7.6. Since the valuation of F1, F7, and F8 are depended on the redesign cost,  $C^a$ , additional vision features for a new platform can be selected by design constraints related to development cost for the features.

We performed sensitivity analysis to investigate the behavior of the estimated option value against chaining system parameters such as the rate of cost saving for product family, the volatility of demand, and the rate of the expected increasing demand. Figure 7.7 shows the effect of the rate of cost saving for product family on

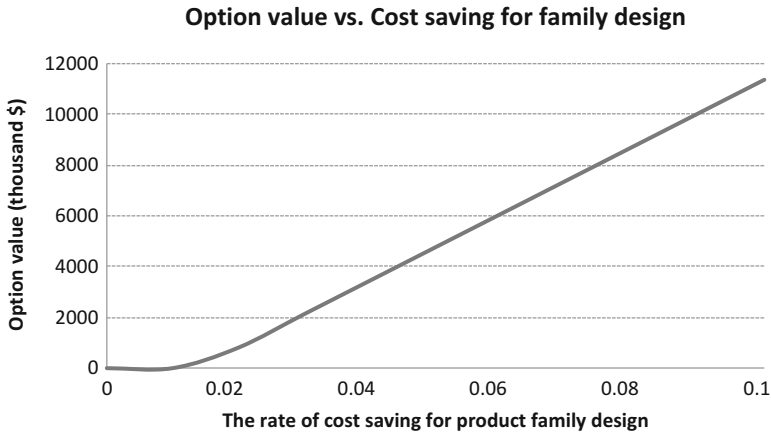


Fig. 7.7 The estimated option value versus the rate of cost saving for family design

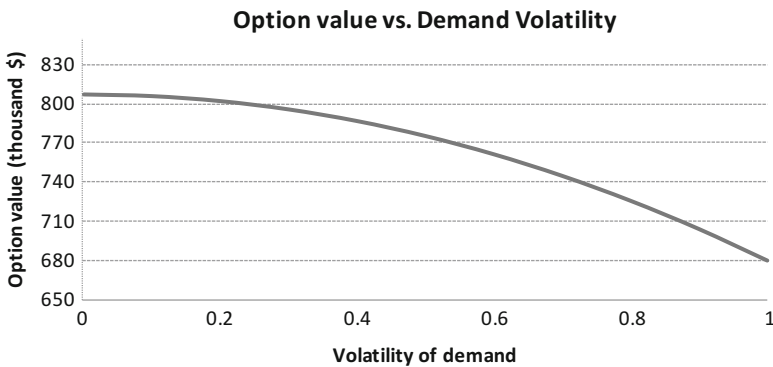
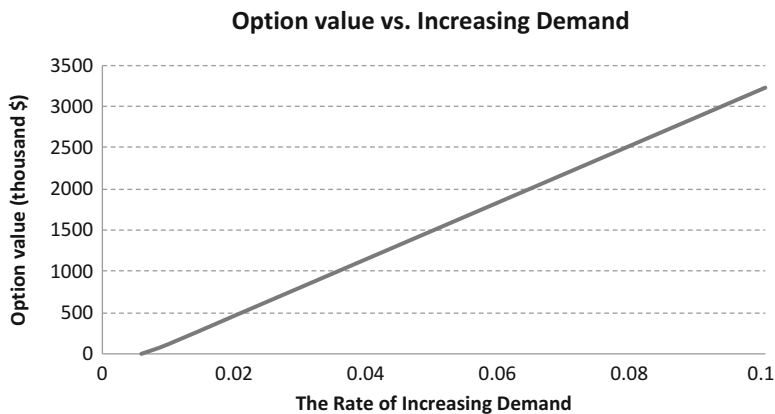


Fig. 7.8 The estimated option value versus volatility of demand

the estimated option value. While the rate of the cost saving increases, the option value increases linearly. Figure 7.8 shows the estimated option value against the volatility of demand. The estimated option value is dropped with an increasing rate while the volatility increases. Figure 7.9 presents the estimated option value versus the rate of the increasing demand that occurs with customers' preferences and demographic trend. Design quality gives positive effects on the option value.

Through the case study, we demonstrate that the proposed valuation model for a module could be used to determine a design strategy for maximizing profits by valuing the module in the family of products. The proposed model can provide a quantitative method to facilitate family design in an uncertain market environment.



**Fig. 7.9** The estimated option value versus the rate of increasing demand

## 7.5 Closing Remarks and Future Work

In this research, we presented a valuation financial model to evaluate design value based on real options analysis in an uncertain market environment. Real options analysis was applied to evaluate the expected worth of introducing new modules as a platform in a product family. Modular product architecture was used to allow a range of trade-offs in determining the specific configuration for a platform at a conceptual design phase. To evaluate and measure design quality of a product, we proposed a preference function using customers' preferences and performance utilities for products.

The proposed financial model can facilitate design strategies that will maximize the expected profit under uncertain constraints, such as demands, demographic trends, and regulations. In a case study, we have applied the proposed model to determine the valuation of accessible modules for a platform in a family of mobile products in an uncertain market environment. We also performed sensitivity analysis to investigate the behavior of the estimated option value against changing system parameters such as the volatility of demand, the rate of the expected increasing demand, and the rate of cost saving for product family.

Since the proposed financial model is focused on modular product families, module configuration issues will be considered as real options analysis for introducing interfaces between modules in product family design. To improve the proposed model, we need to develop a method to better reflect the benefit of family design, social issues, and government regulation. Additionally, since the production cost are sensitive to the valuation of options, future research efforts will be focused on improving production cost models in an uncertain market environment. Also, the proposed method will be compared to other decision-making methods for determining a design strategy in a product family.



## References

- Bollen NPB (1999) Real options and product life cycles. *Manag Sci* 45(5):670–684
- Brach MA (2003) Real options in practice. Wiley, Hoboken, NJ
- Cox J, Ross S, Rubinstein M (1979) Option pricing: a simplified approach. *J Financ Econ* 3(2):229–263
- Cronin MJ (2010) Smart products, smarter services: strategies for embedded control. Cambridge University Press, New York, NY
- Fernandez MG, Panchal JH, Allen JK, Mistree F (2005) Concise interactions and effective management of shared design spaces – moving beyond strategic collaboration towards co-design. In: Proceedings of ASME international design engineering technical conference & computers and information in engineering conference, 24–25 Sept, Long Beach, CA, Paper No. DETC2005-85381
- Ford DN, Sobek DKI (2005) Adapting real options to new product development by modeling the second toyota paradox. *IEEE Trans Eng Manag* 52(3):175–185
- Gamba A, Fusari N (2009) Valuing modularity as a real option. *Manag Sci* 55(11):1877–1896
- Holttta-Otto K, De Weck O (2007) Degree of modularity in engineering systems and products with technical and business constraints. *Concurr Eng Res Appl* 15(2):113–126
- Jiao J (2012) Product platform flexibility planning by hybrid real options analysis. *IIE Trans* 44(6):431–445
- Jiao J, Lim CM, Kumar A (2006) Real options identification and valuation for the financial analysis of product family design. *Proc Inst Mech Eng B: J Eng Manuf* 220(6):929–939
- Johannesson H, Claesson A (2005) Systematic product platform design: a combined function-means and parametric modeling approach. *J Eng Des* 16(1):25–43
- Johnson MD, Kirchain RE (2011) The importance of product development cycle time and cost in the development of product families. *J Eng Des* 22(2):87–112
- Kamrad B, Ritchken P (1991) Multinomial approximating models for options with K state variables. *Manag Sci* 37(12):1640–1652
- Kopin V, Wilbur D (2005) Bayesian serial cost sharing. *Math Soc Sci* 49(2):201–220
- Krishnan V, Zhu W (2006) Designing a family of development-intensive products. *Manag Sci* 52(6):813–825
- Kumar D, Chen W, Simpson TW (2009) A market-driven approach to product family design. *Int J Prod Res* 47(1):71–104
- Longstaff F, Schwartz E (2001) Valuing american options by simulation: simple least-squares approach. *Rev Financ Stud* 14(1):113–147
- Magrab EB (1997) Integrated product and process design and development: the product realization process. CRC, Boca Raton, NY
- Meyer MH, Lehnerd AP (1997) The power of product platforms: building value and cost leadership. The Free Press, New York, NY
- Moon SK, Mcadams DA (2009) A design method for developing a universal product family in a dynamic market environment. In: Proceedings of the design engineering technical conferences and computers and information in engineering conference, Aug 30–Sept 2, San Diego, CA. ASME, Paper No.: DETC2009-86784
- Moon SK, Park J, Simpson TW, Kumara SRT (2008) A dynamic multi-agent system based on a negotiation mechanism for product family design. *IEEE Trans Autom Sci Eng* 5(2):234–244
- Moon SK, Shu J, Simpson TW, Kumara SRT (2011) A module-based service model for mass customization: service family design. *IIE Trans* 43(3):153–163
- Nokia (2008) <http://www.nokiausa.com>
- Ramesh B, Tiwana A, Mohan K (2002) Supporting information product and service families with traceability. *Lect Notes Comput Sci* 2290:353–363
- Rojas Arciniegas AJ, Kim HM (2012) Incorporating security considerations into optimal product architecture and component sharing decision in product family design. *Eng Optim* 44(1):55–74

- Shiau CN, Michalek JJ (2009) Optimal product design under price competition. *J Mech Des* 131(7):071003:1–071003:10
- Shooter SB, Simpson TW, Kumara SRT, Stone RB, Terpenney JP (2005) Toward an information management infrastructure for product family planning and platform customization. *Int J Mass Custom* 1(1):134–155
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf* 18(1):3–20
- Simpson TW, Maier JRA, Mistree F (2001) Product platform design: method and application. *Res Eng Des* 13(1):2–22
- Simpson TW, Siddique Z, Jiao J (2005) Product platform and product family design: methods and applications. Springer, New York, NY
- Smit HTJ, Trigeorgis L (2004) Strategic investment: real options and games. Princeton University Press, Princeton, NJ
- Thevenot HJ, Alizon F, Simpson TW, Shooter SB (2007) An index-based method to manage the tradeoff between diversity and commonality during product family design. *Concurr Eng: Res Appl* 15(2):127–139
- Xiao A, Zeng S, Allen JK, Rosen DW, Mistree F (2002) Collaborating multidisciplinary decision making using game theory and design capability indices. In: Proceedings of the 9th AIAA/ISSMO symposium on multidisciplinary analysis and optimization, 4–6 Sept, Atlanta, GA
- Zacharias NA, Yassine AA (2008) Optimal platform investment for product family design. *J Intell Manuf* 19(2):131–148

**Part II**  
**Platform Architecting and Design**

# Chapter 8

## A Proactive Scaling Platform Design Method Using Modularity for Product Variations

Keith Hirshburg and Zahed Siddique

**Abstract** To be competitive in the current business environment, a company or engineering firm must be able to produce new products or designs in the marketplace with better quality and greater customization than both their national and international competitors. These business entities must also be able to accomplish this at a more strenuous pace than their competitors to capture the largest market share. In this chapter, a scaling, small product, *proactive platform design method using modularity* (PPM) for product variations is presented to assist the company or firm in achieving the highest competitive result. In Chap. 30, we also present a case study to demonstrate how this method can be effectively instituted in a proactive product design. Even though this method and case study are directed to small product family development, any product family design with commonality can benefit from using these ideas to improve the design process.

### 8.1 Introduction

To bring a set of products to the market in an intelligent and economical way, a company must use an orderly and defined process to design and manufacture these products. Companies are striving to deliver greater quality, more varieties, faster response, more innovative designs, and lower prices (Bower and Hout 1988; Stalk and Hout 1990). New models are introduced in the market more frequently, while the number of mass-produced models is decreasing (Schile and Goldhar 1989). Although different researchers (Bower and Hout 1988; Hirsch and Thoben 1997; Hollins and Pugh 1990; McDermott and Stock 1994; Wheelwright

---

K. Hirshburg • Z. Siddique (✉)  
School of Aerospace and Mechanical Engineering, University of Oklahoma,  
Norman, OK 73019, USA  
e-mail: [zsiddique@ou.edu](mailto:zsiddique@ou.edu)

and Clark 1992) have highlighted different reasons for family of products, there is a consensus that for companies to survive in the current global market, they must move towards a platform-based customization or family of products.

Companies are being faced with the challenge of providing as much variety as possible for the market (external) with as little variety as possible between products (internal). One of the key elements in product family is the product platform. "A product platform is a collection of the common elements, especially the underlying core technology, implemented across a range of products" (McGrath 1995). One way to achieve this is by developing the product platform carefully and then using different modules to provide product variety. This approach requires configuration rationalization of the platform and the product family. Configuration design involves determining which modules are in the product, what are the components in the modules, and relationships among the components and modules. A well-defined product platform is required to support family of products. The approach advocated in this chapter, and by many strategic marketing/management researchers and designers/engineers alike, is to design and develop a family of products with as much commonality between products as possible with minimal compromise in quality and performance.

Focusing product strategy at the platform level simplifies the product development process because there are fewer platforms than products and major platform decisions are only made every few years. "A clear platform strategy leverages the resulting products, enabling them to be deployed rapidly and consistently" (McGrath 1995). A platform approach encourages a long-term view of the product strategy. Implementing commonality to develop platforms for a set of similar products requires product configuration reasoning to determine the product platform and then to identify the portfolio associated with the platform.

Review of the traditional individual product engineering design process reveals an insufficiency, that is, the current process does not take into account reusing the design or parts of the design to create other similar products or solutions. Currently in the last few decades, as companies became more competitive, and the markets became more segmented, the product designers need an efficient approach to design a family of products. This method requires a company to design products at a more aggressive pace, provide more variety to customers, while increasing design and manufacturing efficiency and reducing financial burden on the company. Consequently, the overall objective of this chapter is to develop a method for product family design through using a scalable platform and modular product family while using concurrent teamwork, human-involved design negotiation with mathematical optimization, and design for manufacturing. We also investigate changes and extensions to existing individual product design processes to accommodate design of product families.

## 8.2 Related Work

A product family is a general group of related products that are created off of single or multiple platforms (Simpson et al. 2006). Put another way a product family is an approach “to obtain the biggest set of products through the most standardized set of base components and production processes” (Siddique and Rosen 1999). The product family attempts to fulfill most or all of the customer niches in a certain market(s). Examples of product families range from Black-N-Decker drills to light SUVs made by the automobile manufacturers. Other terms associated with product families are product line and product group. Product family design can be described as “a conceptual structure and overall logical organization” that is used to create a family of products through utilizing commonality achieved from providing a “generic umbrella” to extend product line structure (Jiao et al. 2007).

A product platform can be defined as a set of common components, modules, processes, and assets from which a stream of derivative products can be efficiently developed and launched (Meyer and Lehnerd 1997; Robertson and Ulrich 1998). Extended further the platform can be identified in physical and nonphysical terms, meaning actual knowledge can be considered a platform instead of just designed components. In this chapter a platform will be considered as the common components, modules, and interfaces that are involved in driving the performance of a family.

A proactive platform is a top-down approach, meaning that a company strategically creates a product line “based on a platform and its derivatives” (Simpson et al. 2006). A company performs proactive platform design by designing the platform components from the beginning to work in conjunction with each other that makes a base for the product line. This design would not try to create a platform from existing products or components. An example of a top-down approach is the design of the product family of Walkman, created by Sony Corporation (Sanderson and Uzumeri 1997). A reactive design, the bottom-up approach, is a platform design process where a company takes an existing set of distinct products and tries to consolidate them into using a single set of components (Simpson et al. 2006). Currently, there is a need for a proactive approach to product family design that can utilize tools that have already been developed by researchers.

Modular architecture is a product architecture defined by a one-to-one or one-to-multiple construction of functional elements (Simpson et al. 2006). In product family there are three main types of modularity: functional modularity, technical modularity, and physical modularity. Integral product architecture is defined by having a complex or coupled mapping of functional elements to physical structures or interfaces (Simpson et al. 2006). A scale-based or parametric product family is based on a platform that stretches or shrinks to create different products in the product family. This stretching or shrinking can be a single component of the platform, a few components, or even the entire platform to create the different products of the family. GAM (Lu and Zuhua 2006) is a method for designing a platform through the use of a genetic algorithm to find the components of the platform from a list of design variables.

Design affordance is a design process of allowing and disallowing certain conditions, variables, and aspects, when conducting a design process. Original design of a complex system starts from the view at the system level and then is decomposed in subsystems and finally down to the component level. Design affordance seeks to understand the system as a whole in terms of functions, interactions, reactions, and even emotions. Affordance means “what it provides, offers, or furnishes to a user or to another product” (Maier 2008). The product family is designed by limiting or boosting the affordances. Gonzalez-Zugasti et al. (2000) present a method for designing a product family through the use of an interactive team-based negotiation of components. This method’s inputs are (1) the requirements for the product family, (2) the variables, (3) whether the variables are common, (4) and the interrelationship of the variables. This method computes a platform using basic optimization and then the designer can create variants of this platform to suit the individual product needs. Varieties in many products are based on functionality; hence, an approach is needed that uses modularity as a mean to support family products.

### 8.3 Proactive Platform Design Method Using Modularity

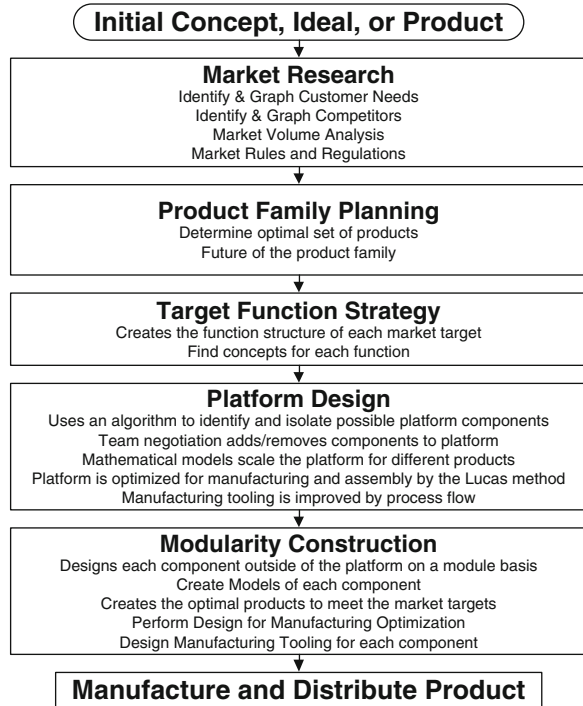
The proposed PPM design method for creating a scaling, small product, and proactive platform, using modularity for product variations, is comprised of five steps to conceive a product family from an initial concept. The steps are (1) market research, (2) product family planning, (3) target function strategy, (4) platform design, and (5) modularity construction. These five phases are shown in Fig. 8.1.

#### 8.3.1 Step 1: Market Research

Market research and analysis for a potential product is necessary for a design team to be aware of all the metrics concerning a product’s performance and the importance of these features to offer an optimized product line to the customers (Stobart 1994). Distributing a product line that fails to meet the customer demands will create a lower than desired sales volume in the present and will also “negatively affect future sales” for the company by damaging the company’s brand name (Stobart 1994). Associated sub-steps are:

*Identify customer needs:* To be able to determine the customer needs, the design team will need to research the different market segments surrounding the product line’s main purpose or function. The market segment can be designated by viewing existing products that relate to the product line’s base function and determining logically where the product lineup fits. The determination of the market segment can be performed by thinking broadly and working towards a more defined market segment.

**Fig. 8.1** Outline of the design method



The result of narrowing down the market segment leads to a more defined set of customer demands, but the consequence to narrowing the market segment will also reduce the sales volume for the product line. A trade-off must be decided between achieving a large amount of total sales and the ability of the products in the product line to fit the needs of a specific set of customers. After narrowing down from a broad market focus, a preliminary main customer segment group should be chosen and the design team should begin accumulating information on the customers of these common segments. This can be accomplished by performing a review of the customers and a review of the competition in these segments. A review of the customers of the market segment is difficult, time-consuming, and statistically non-exact process (Mendenhall and Sincich 2007). To determine the customer needs directly, the team can interview the customers in person or over the phone. A less direct interview can be accomplished through feedback of company's previous products, Internet polls, or email polls. Direct interviews provide less biased information due to people having to put forth less effort and reason to give the company information (Campbell 1974). The direct interview methods usually result in a normal curve for replies if the polls have a large enough population (Mendenhall and Sincich 2007). Less direct feedback methods result in a greater amount of bias. These biases result from the feedback volunteered by customers who appreciate the product in the upper extreme or who dislike the product in the lower extreme.



This indirect feedback method usually results in a reverse normal curve (Mendenhall and Sincich 2007). The indirect feedback method with a large population highlights what features are required in the product family and what features should be left out. The interviewee rates each feature on the interview form with a scale that ranges from the feature that is not important in consideration for buying the product to the feature being extremely influential on the purchase. The information can be organized visually and analyzed to determine the importance of the features. Two graphs should be created, one for the direct interview method and one for the indirect polling method. By overlapping the two graphs, the design features that are most important in the product offerings can be determined by identifying repeated peaks.

*Identify competitors:* Competitors first need to be identified (Clark and Montgomery 1999) to perform a review of how the competition fulfills the market demand. This can be accomplished by identifying products offered in the market segments and then researching the product features, the quality of the features, how the product performs these features, and how many of these products are purchased. For each product being researched, consumer/user evaluation is gathered using a survey. The consumer/user survey uses four categories to evaluate features:

- Metric A determines if the competitor's product includes the selected feature.
- Metric B describes the quality and durability of the selected feature.
- Metric C is the performance metric and is used to gauge how well the feature performs in the product.
- Metric D is the population metric and is the number of products that are sold per year by the specific competitor.

Metrics A through D are used to calculate the valuation of the feature from the product and are shown in Eq. (8.1).

$$\text{Feature Value} = A_{\text{feature}} * \frac{B_{\text{quality}} - 1}{4} * \frac{C_{\text{performance}} - 1}{4} * D_{\text{amount}} \quad (8.1)$$

After all the competing products in the market have been evaluated, the calculated feature values are compiled to visualize how the competing products match the customer demands, using Eq. (8.2).

$$\text{Total Feature Value (feature}_x) = \sum_{n=1}^i (\text{feature}_x)_i \quad (8.2)$$

Using the compiled scores of the features, the market demands can be compared to the features offered by the competitors. Statistical methods can be applied to gain a more defined edge over the competition in the market segment. After the primary market segments have been researched, the team should decide if reaching market segments outside of the selected ones can benefit revenues, without degrading the product line. These outer market segments might be incorporated into the product line by leveraging a variation of the platform. The outside markets could be

potentially reached by the initial product offering or by a future variation of the platform released at a later date. Investigation into other segments can be accomplished by repeating the same process of research in the main segments.

*Market volume analysis:* Market volume analysis can be used to determine the number of products needed to meet the customer demands and the number of products currently on the market. Volume analysis should not only cover the total amount of products to be sold but also find the relationship between price and number of units that can be sold at that price. The market volume analysis is used to help decide how many and the price of units for each variation.

*Rules and regulation bodies:* Along with the laws and regulations involving patents and other information about the designs, there are also laws regulating how these products perform. Each market segment will have sets of regulations that it may have to follow. Some examples of regulatory bodies are the EPA and OSHA, but there are many other agencies and regulations that may need to be researched and followed. All rules and regulations should be researched for each market segment for the product line, before the product is designed. Proactive research in this area will have a large cost reduction, compared to researching retroactively.

### **8.3.2 Step 2: Product Family Planning**

The objective of product family planning is not just to plan an optimal product line for the instance it is released but also to plan out the future offerings, variations, and upgrades to the products, until the next generation of the platform product family can be released. The associated sub-steps are described next.

*Optimal set of products:* To create an optimal set of products, the team must first convert the product qualitative features of the marketing process into quantitative features the product can be designed from. This is accomplished by converting the generic qualitative feature into a measureable quantitative feature. The conversion can be accomplished by using well-established methods such as quality function deployment. The optimal set of products, also known as the market targets, can be created by using a specific selection of quantitative features needed to create the performance demands of the target. This set of products should fulfill all of the major requirements in the market segments and should be the products the team is attempting to produce in the product family. If the amount of features and details of the features become too complex, the optimal product line can be selected by a genetic optimizing algorithm.

*Future of the family:* The success of the product family depends on how it is leveraged over time until the next generation of the product family can be designed and manufactured. A large part of the planning for the future is determining the time length between product family generations. This time between generations is dependent upon the market competitor's generational length and the capabilities

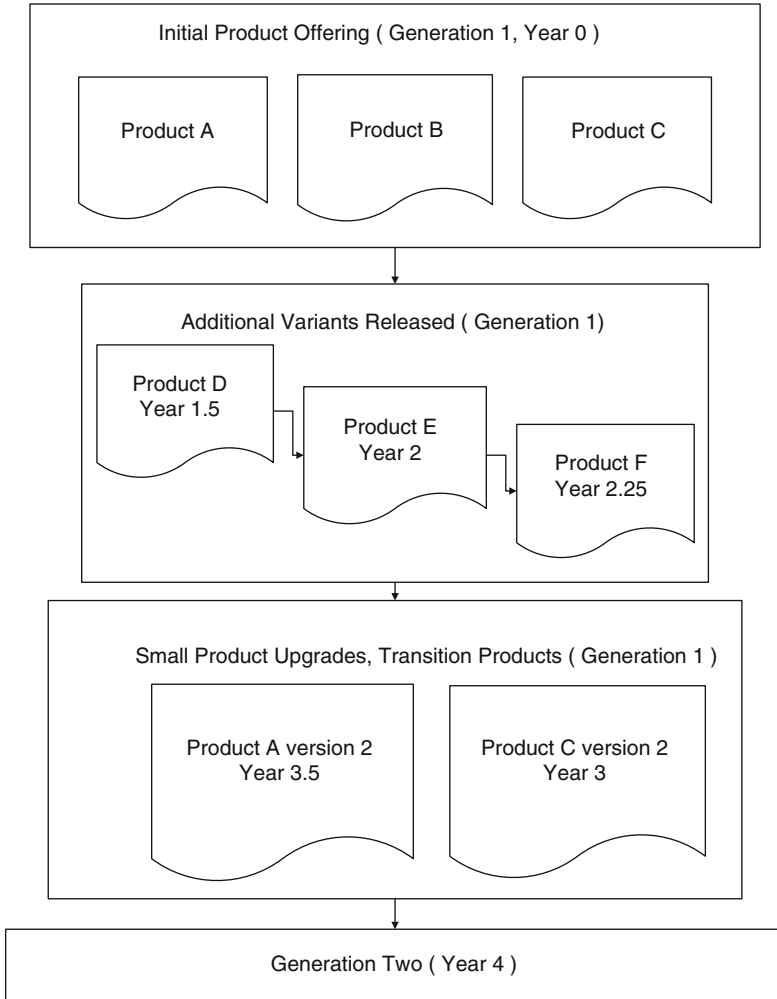
of the design team's company. Product life cycles can be as long as 10 years in the example of Sony and its PlayStation systems or as short as a year as is found in some cell phone product families. To find the market average for the product life cycle, the team will need to research when each product of the competitor's was first released and when the product was replaced by the next generation of product. If the product gets replaced by a product, with only a small amount of upgrades, it should not be considered a new generation. After finding the average product life cycle of the competitors, the team should plan on matching this life cycle or even trying for a longer product life cycle. Longer product life cycle usually results in product line generation being more profitable. In certain cases sales will trail off for the longer the cycle because of market saturation or newer models from competitors being available in the market. Falling sales are due to the product line not fulfilling the customer needs as well as the competitor's new products, and this can lead to a damaging of the companies brand name. After the product life cycle has been determined, the product release dates should be decided upon to keep the customer segment interested in the company, to allow for the marketing department to have new products to market, and to provide transition time between the generations. A basic example of a planned time leveraging of a small product family is shown in Fig. 8.2.

### ***8.3.3 Step 3: Function Strategy***

The function strategy is the road map for the product family design process and is composed of function structures for each of the products in the family. The first step is to create function structures for each of the market targets to provide an outline of the functions included in each product. A function structure is the mapping of the different flows of material, energy, and information within a design. A full creation of a function structure should be a road map to demonstrate what needs to be designed. Even though it presents what needs to be designed, it does not provide how the product should be designed. For each process defined in the function structure, the process can be extrapolated into a component(s) of the platform or a modular component(s) in the product variants. At this stage, each process should be isolated and named for use in both the platform design process and the non-platform design process.

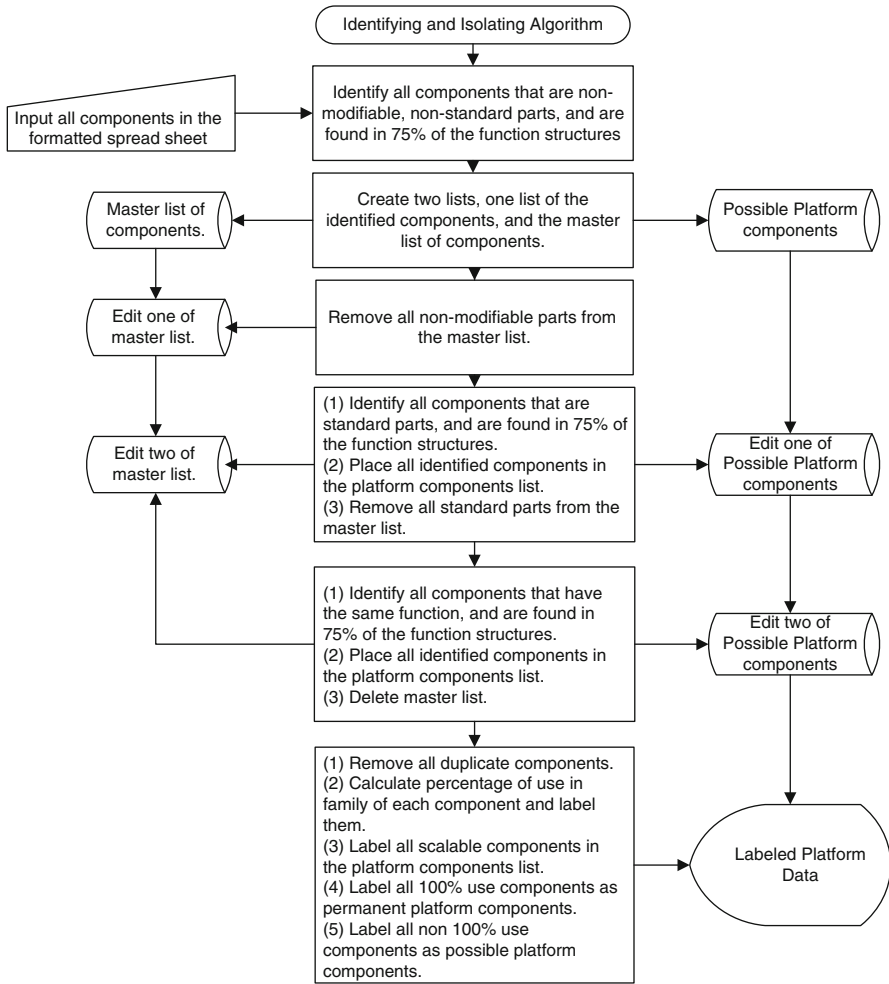
### ***8.3.4 Step 4: Platform Design***

All metrics that can be used to describe the performance of a product family are influenced largely by a base number of components, known as the platform. Choosing an optimal set of components and then optimizing those components need to be accomplished to produce a product line that is successful in the current competitive environment. Sub-steps are:



**Fig. 8.2** A time-leveraged example of a generic product plan

*Determining platform components:* To determine platform components, the function structure needs to be broken down for each of the market targets into its components, which are placed in a matrix. This matrix logs the functions in rows and places the details of the function in columns under component name, market target, component function, scalable, standard part, and modifiable. After all these inputs are entered into the matrix, an algorithm (see Fig. 8.3) identifies and isolates all possible platform components. This is accomplished through the use of multiple processes. Components that do not have the flexibility to be modified without adversely affecting the performance of the product are considered nonmodifiable. The first process of the algorithm identifies all components that are nonmodifiable



**Fig. 8.3** Identifying and isolating algorithm

but found in 75 % (this percentage can be increased or decreased based on the product. A higher percentage will result in lower commonality) of the market targets and places these components into the platform with removing the leftover nonmodifiable components from being considered for the platform. The second process identifies all components that are off-the-shelf/standard parts and are found in 75 % of the market targets and isolates them. The process then places these components into the platform and removes the remaining standard parts from being considered for the platform. The third process identifies all the components that share a function with at least 75 % of the market targets. These identified components are isolated and placed into the platform consideration components

and remove all the function components that did not meet this percentage. The last process evaluates the collection of platform components and removes the duplicates and then calculates the exact percentage use in the targets for each component. After this evaluation, the process tags all components with 100 % use as permanent platform components and labels the rest with a possible platform component tag. Finally the process highlights all scalable components for easy identification.

After the use of the algorithm, the team will need to decide which of the possible platform components should be included in the platform, and this can be done by using a negotiation model. After the algorithm and the team negotiation, the platform components are confirmed and the team can continue to designing, modeling, and optimizing the platform.

*Platform design modeling:* The platform has been selected and now it needs to be designed and modeled. The platform components should be broken into two categories: components that will be scaled and components that will not be scaled. The design entails creating models using the requirements and functions. In the design of scaled components, the restrictions should be modeled using both the maximum and minimum dimensions required to produce the maximum and minimum performances. These multiple dimensional models are designed so the non-scaled components in platform and the non-platform components can be designed within potential space restrictions. The designers should design the scaled component's CAD models with true flexibility for on-the-fly changes. Non-scaled connections must be designed to remain static during the on-the-fly changes to allow for connectivity in all scales. The static connections to the non-scaled components will allow for modularity in the connections for the non-scaled and the non-platform components.

*Platform optimization:* Platform optimization is the modifying of the scales, dimensions, and properties of the platform components to achieve the desired performance. The optimization of the scaled platform components can be accomplished using existing optimization approaches to determine the optimal set of scales (Simpson 2004). For the non-scaled components, the use of FEA and CFD analysis should be all that's needed. The designers should repeat platform design modeling and scaling (Fig. 8.4) for different scaling components and modules.

*Design for manufacturing improvement on the platform:* Improving the platform for manufacturing is important since the platform will be manufactured for each of the products produced. Any waste or underperformance will be repeated many times over, so any improvement found from being thorough in the design and redesign of components for manufacturing is very important. The manufacturability and assemblability of the platform components are improved by following the Lucas method, which is a method based on a difficulty value assessment instead of recording time of assembly as in the Boothroyd and Dewhurst method. The Lucas method assigns values to three processes: functional analysis, feeding analysis, and fitting analysis which leads to an assessment on the assembly of the product. The efficiency of the design is analyzed through functional analysis by providing a

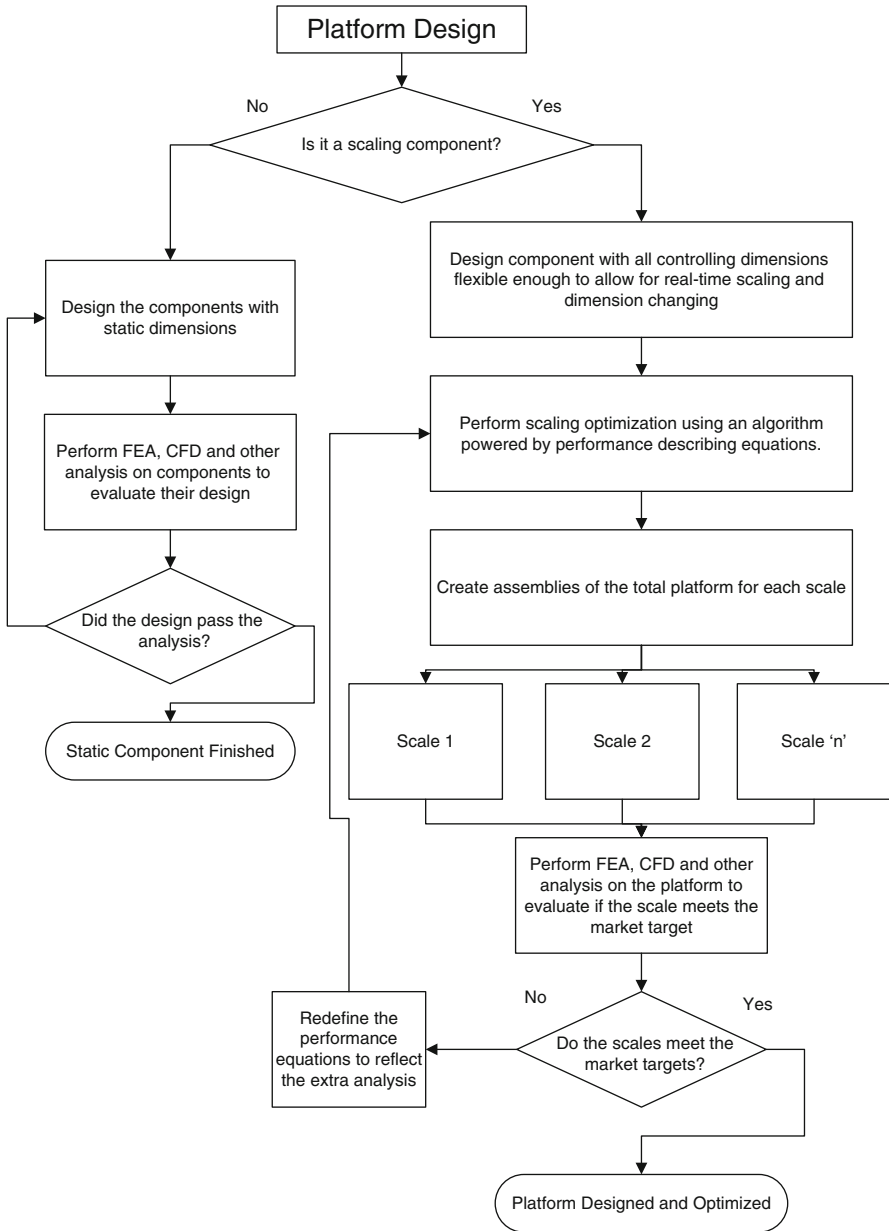


Fig. 8.4 Platform design modeling and scaling flow process

percentage of essential components of the product (parts that are deemed to be essential to the product's function; parts that are not essential to the product's function include fastening and locating). The calculation of functional analysis is shown in Eq. (8.3):

$$\text{Functional Efficiency} = \frac{\sum \text{essential components}}{\sum \text{essential components} + \sum \text{non essential components}} \quad (8.3)$$

The handling of components, during assembly, is analyzed using feeding analysis and is shown in Eq. (8.4). In the feeding analysis, the problems associated with the handling of the part are scored using an appropriate table. For each part, the individual feeding index is scored. Similarly fitting ratio is calculated [Eq. (8.5)].

$$\text{Feeding Ratio} = \frac{\sum \text{Part Feeding Indices}}{\sum \text{essential components}} \quad (8.4)$$

$$\text{Fitting Ratio} = \frac{\sum \text{Part Fitting Indices}}{\sum \text{essential components}} \quad (8.5)$$

The fitting ratio is used to analyze the insertion of the component into the products during assembly. The Lucas manufacturing analysis is an assessment on the complexity and cost of the manufacturing by performing analysis on many different metrics. The manufacturing cost is calculated using Eq. (8.6):

$$\text{Manufacturing Cost Index} = C_C * C_{MP} * C_S * (C_T \text{ or } C_F) * P_C + V * C_{MT} * W_C \quad (8.6)$$

where  $C_C$  = complexity factor,  $C_{MP}$  = material factor,  $C_S$  = minimum section,  $C_T$  = tolerance factor,  $C_F$  = finish factor,  $P_C$  = processing cost,  $V$  = volume (cubic millimeters),  $C_{MT}$  = material cost,  $W_C$  = waste coefficient.

The Lucas method uses seven steps to improve the manufacturability and assembly of the platform. Step one involves the specification of the platform, step two is the design of the platform, step three is the functional analysis, step four is the feeding analysis, step five is the fitting analysis, step six is the assessment of the assembly analyses, and step seven is the assembly analysis.

*Manufacturing tooling design:* Since the platform is the backbone of the product family, the platform design must incorporate the manufacturing tooling. The platform is the largest contributor to product quality and performance, and it is advisable to have the platform manufactured in-house to have better control over the manufacturing. The only two exceptions to in-house manufacturing of platform components would be fasteners and other off-the-shelf components or with using an experienced partner that is committed to manufacturing the platform to the exact



design specifications. A great reduction in manufacturing time and considerable tooling cost can be achieved by improving the tool passes. A tool pass operation that is not optimized leads to wasted time, incorrect tolerances, and accelerated tool wear. Optimizing a machining operation involves optimizing the machine time and tool wear for volume, surface, and finishing operations.

Dies for injection molding should be designed with easy access to the created part, low amount of lost material in the mold, and proper cooling to allow for repeated moldings and to create parts with near-optimal tolerances to eliminate the need for post-processing. The process for optimizing an injection molding, molding, and forging operation involves six assessments. The six assessments are the following: does the die provide easy access to the part, is excessive material wasted in the die, does the design have proper cooling, can the component be created with better tolerances, and does the part need post-processing?

*Nonphysical component platform:* The nonphysical component platform is the sharing of guidelines or styling quos throughout the product lineup. A well-defined platform will involve multiple nonphysical platform entities. The nonphysical platforms can be reused in future products for the company to make the design processes easier. The reuse of nonphysical platform can lead to having the products carry a distinct look that adds to the brand identity. A very important non-component platform item is the fasteners. All fasteners should be of the same type, and if at all possible, use the same tool for assembly and reassembly after maintenance or repair work will be more efficient.

### **8.3.5 Step 5: Modularity Construction**

Modularity construction is a phase to design the non-platform components of the product line. The use of modularity in the design of the products leads to the ability to provide product variations of the product line.

*Module design of components:* Modularity is the one-to-one mapping of the connections used by components in a product. This requires all non-platform components to use modular connections when connecting to the platform and to other components. To allow for greater customization in the design of the product, each component should be designed using only its particular details and not the details of the whole product. The particular details would include the input, output, and behavior of the component instead of the requirements of the entire product. Designing the components to work with a particular interface and a set of low-level details will allow for greater customization and the ability to upgrade single components without being forced to redesign the entire product.

*Component design modeling:* Component design modeling is a restricted single-product design process. The restrictions on the design process are the behavior of the component from the function structure, the interface of the component, and the

sizing constraints derived from the platform and market research. The specification definition for Ullman's design process would not be the entire product, but the purpose and behavior of the component being designed (Ullman 2002). The customers of the component are all components that will be interfacing with it. The customer requirements of the components are what the component is required to perform and the restrictions on how the component can perform. The competitors of the product are the similar components produced in other company's products. The competitor's should be evaluated for their positives and negatives. The design process of the components of the different product variants is found in Fig. 8.5. The component design process is repeated for each different component in the product line that is not a platform component.

*Creating the market targets:* With the components designed, the market targets can be created by selecting a scaled platform and adding certain components to produce the necessary performance. If the created product's performance does not match the market targets, the integration of the components needs to be checked for correctness and the underperforming components need to be identified. The underperforming components may need to be redesigned, or it means reaching the market targets is not possible within the product constraints.

*Manufacturing improvements for optional components and products:* The manufacturability and assembly of the product are improved by following a modified Lucas method that is applied to each component to the product. The Lucas method is modified by performing the Lucas manufacturing analysis after each step of the assembly analyses. The modification of the standard Lucas method disallows an extreme removal of components and lessens the chance of complexity in the components. The modified Lucas approach (Fig. 8.6) validates a component design or forces a component redesign that is optimized for assembly onto the platform or onto other components.

*Component manufacturing tooling design:* In most cases the small products will be made using multiple materials, with a mix of different components. Some components will be manufactured in-house and some components will be outsourced. For in-house manufactured parts, the tooling design is important to reduce manufacturing waste.

*Production volume costing:* There are three major factors to the cost of manufacturing the product: labor cost, material cost, and machine cost. Making these factors as cheap as possible will in turn make the product more profitable to produce and cheaper for the consumer to purchase. This is a feedback loop that contributes to the economic success of the product line. To lower the material costs, the manufacturing of the components should be designed to waste as little material as possible and to create as few defects as possible. If the manufacturing facility is located in a cheap area of labor, emphasis should be placed on accuracy and quality of manufacturing instead of automation. If the cost of labor is high, the manufacturing process should include as much automation as possible. The cost of production also includes several types of indirect costs known as overhead.

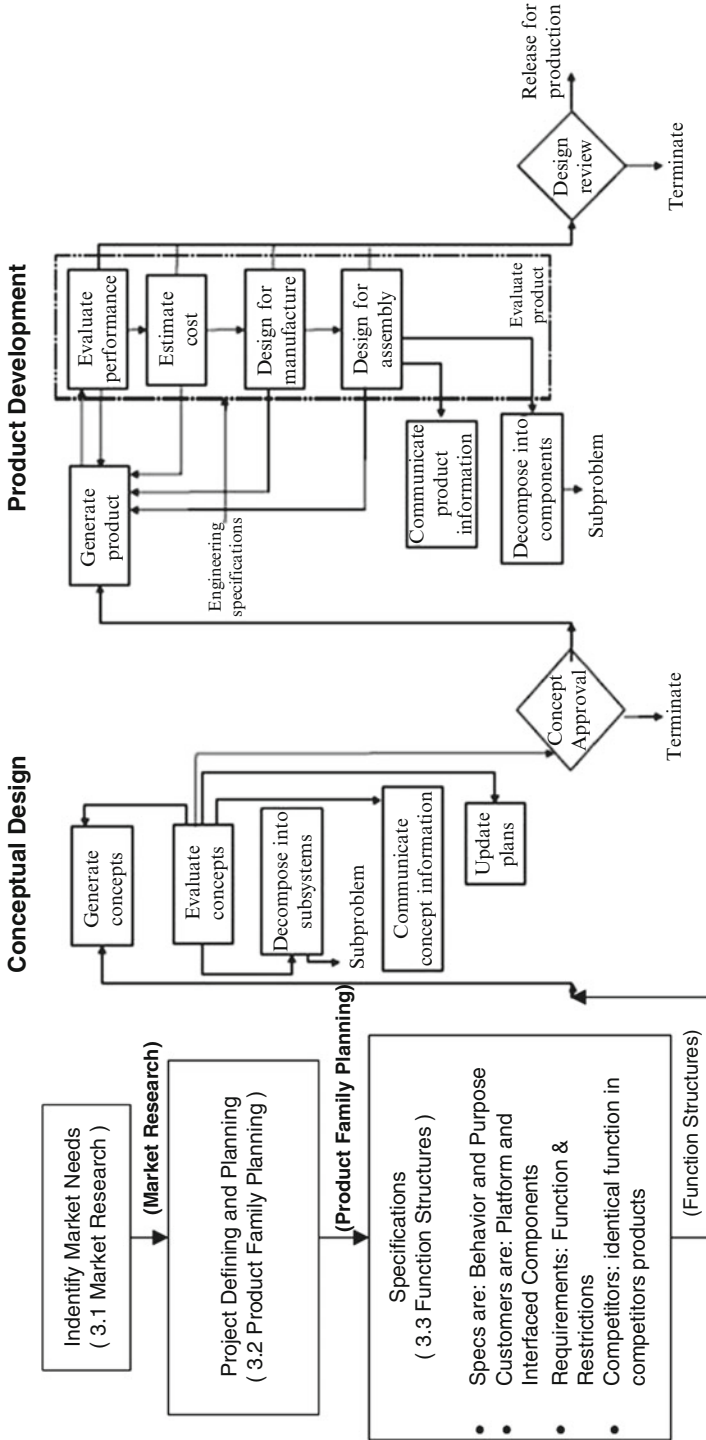


Fig. 8.5 Modified version of Ullman's design process used to create the product components

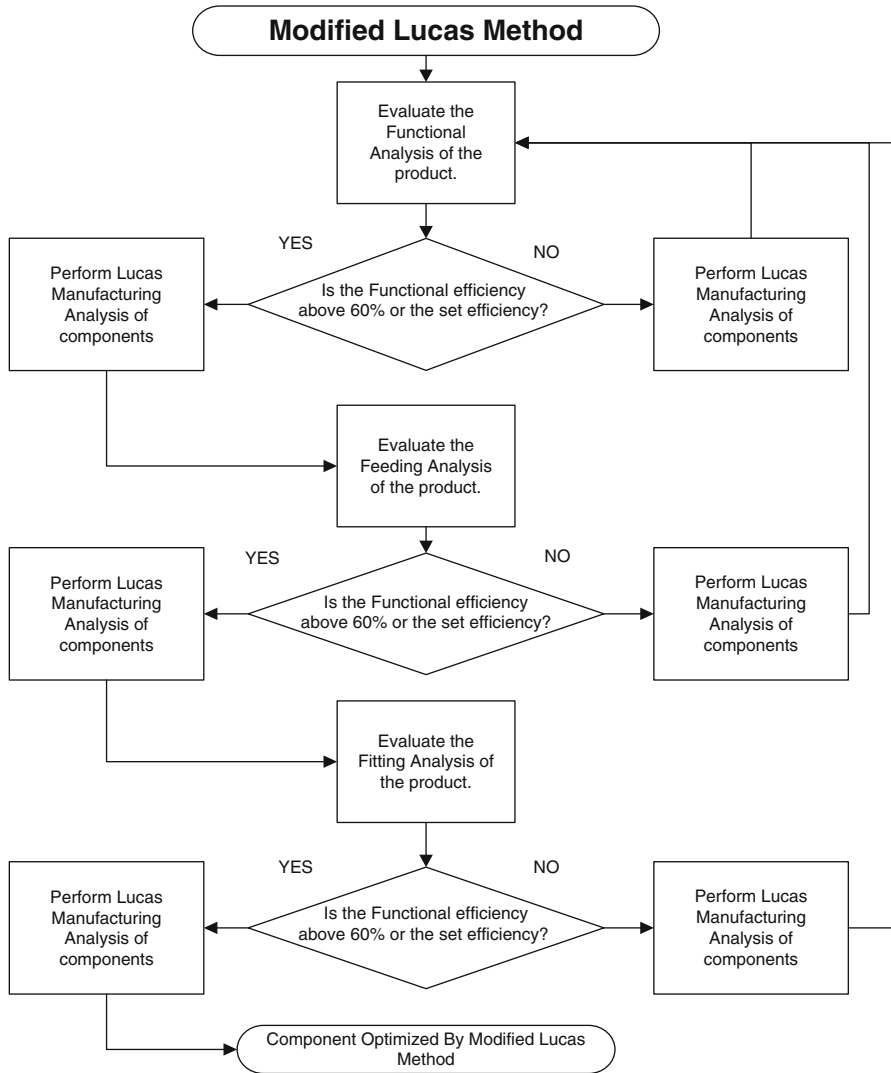


Fig. 8.6 Process flow chart for the modified Lucas method

Examples of overhead include permits, building maintenance, design costs, and marketing costs. Total cost to manufacture a product line is the addition of material cost, labor cost, machine cost, and indirect cost. Total cost to manufacture a product line is:

$$\begin{aligned}
 \text{Total Cost} = & \sum \text{Material Cost} + \sum \text{Labor Cost} + \sum \text{Machine Cost} \\
 & + \sum \text{Indirect Cost}
 \end{aligned}
 \tag{8.7}$$

Cost per product can be calculated as:

$$\text{Cost per Product} = \frac{\sum \text{product}_A}{\sum_{i=0}^n \text{product}_i} * (\text{Total Cost}) \quad (8.8)$$

## 8.4 Concluding Remarks

The presented design method is for a scaling, small product, proactive platform design, using modularity for product variations. The method begins with performing market research through direct and indirect customer interviews and also through evaluations of current competitor's products. The method then takes the results from the research and provides steps for planning of the whole product generation from initial offerings to variants and finally updates of the products. With the products planned, the method is used to create function structures to describe each of the products and to identify the components needed to create the functions. The method then utilizes an algorithm to identify and isolate components common in all or most of the structures and then uses team negotiation to select and design a platform for driving the entire product family. The method also highlights the Lucas method as an optimizer for both assembly and manufacturing of the platform, suggesting a means for optimizing the manufacturing tooling of the platform. Following the design of the platform, the next step is to design the non-platform components using David Ullman's mechanical design process in a modular mapping to drive variation and ensure ease of integration. Along with the design of the non-platform components, the Lucas method and manufacturing tooling design are used to improve the product. Using the discussed steps, a design team can proactively create a viable and competitive product family offering.

In Chap. 30 a family of landscaping blower vacuums case study is presented using the described design method. The case study provides an example of both direct and indirect interviewing for the market research and an example of a competitor analysis of the market segment using both Stihl and Echo competitors. Furthermore, the case study provides analysis on what the customer base demands and created market targets and product variants to meet these demands. The case study concludes with CAD models provided for the main market target products.

## References

- Bower JL, Hout T (1988) Fast cycle capability for competitive power. *Harv Bus Rev* 66:110–118  
 Campbell SK (1974) *Flaws and fallacies in statistical thinking*. Prentice Hall, Englewood Cliffs, NJ  
 Clark BH, Montgomery DB (1999) Managerial identification of competitors. *J Market* 63:67–83

- Gonzalez-Zugasti JP, Otto KN, Baker JD (2000) A method for architecting product platforms. *Res Eng Des* 12:61–72
- Hirsch B, Thoben KD (1997) Why customer driven manufacturing. In: Wortmann JC, Muntslag DR, Timmermans PJM (eds) *Customer driven manufacturing*. Chapman and Hall, New York, pp 33–44
- Hollins B, Pugh S (1990) *Successful product design*. Butterworths, Boston, MA
- Jiao J, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18:5–29
- Lu Z, Zuhua J (2006) A genetic algorithm for scale-based product platform planning. In: Jiao L et al (eds) *ICNC. LNCS, vol 4221*. Springer, Berlin, pp 676–685
- McGrath ME (1995) *Product strategy for high- technology companies*. Irwin Professional Publishing, New York
- Maier JR (2008) Rethinking design theory. *Mech Eng* 130:34–37
- McDermott CM, Stock GN (1994) The use of common parts and designs in high-tech industries: a strategic approach. *Prod Inv Manage J* 35:65–68
- Mendenhall W, Sincich T (2007) *Statistics for engineering and the sciences*, 5th edn. Prentice Hall, Upper Saddle River
- Meyer MH, Lehnerd AP (1997) *The power of product platforms: building value and cost leadership*. Free Press, New York
- Robertson D, Ulrich K (1998) Planning for product platforms. *Sloan Manage Rev* 39:19–31
- Sanderson SW, Uzumeri M (1997) *Managing product families*. Irwin, Chicago
- Schile T, Goldhar JD (1989) Product variety and time based manufacturing and business management: achieving competitive advantage through CIM. *Manuf Rev* 2:32–42
- Siddique Z, Rosen DW (1999) Product platform design: a graph grammar approach. In: *ASME design engineering technical conference*. ASME, Las Vegas, NV, pp 1–12
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf* 18:3–20
- Simpson TW, Siddique Z, Jiao J (2006) Platform-based product family development. In *Product platform and product family design methods and applications*. Springer Science, New York, pp 1–15
- Stalk GJ, Hout T (1990) *Competing against time*. Free Press, New York
- Stobart P (1994) *Brand power*. NYU Press, New York
- Ullman DG (2002) *Mechanical design process*. McGraw Hill, Columbus
- Wheelwright SC, Clark KB (1992) *Revolutionizing product development: quantum leaps in speed, efficiency and quality*. Free Press, New York

# Chapter 9

## Architectural Decomposition: The Role of Granularity and Decomposition Viewpoint

Katja Hölttä-Otto, Noemi Chiriac, Dusan Lysy, and Eun Suk Suh

**Abstract** Before any platform development, one must create the representation of the products' architectures. Typically, one would start by decomposing the existing or proposed systems into smaller subsystems or modules. This is a critical step since the remainder of the platform development will depend on the choices made at the decomposition phase. This chapter will discuss how to decompose a product architecture. Specifically we will address the decomposition choices such as level of granularity and different decomposition viewpoints and how they affect the final resulting architecture.

### 9.1 Introduction

Architectural decomposition is the decomposing of a product or a set of products into smaller subsystems and eventually components. These lower-level subsystems and their interactions form the product architecture. This is the common starting point for all of product platform development, specifically for module-based platform development. For example, Fellini et al. (2006) discuss how the commonality choices for product families are done using a step-by-step process, where after

---

K. Hölttä-Otto (✉)

Engineering Product Development, Singapore University of Technology and Design, Singapore 138682, Singapore  
e-mail: [Katja\\_Otto@sutd.edu.sg](mailto:Katja_Otto@sutd.edu.sg)

N. Chiriac

Department of Mechanical Engineering, University of Massachusetts Dartmouth, North Dartmouth, MA, USA

D. Lysy

Xerox Corporation, Webster, NY, USA

E.S. Suh

Department of Industrial Engineering, Seoul National University, Seoul, South Korea

designing the individual products, the second step is identifying the components that can be shared. These components are results of a decomposition. Similar choices are needed before platform concept evaluation (Hölttä-Otto and Otto 2005).

Further, Thevenot and Simpson's (2005) method to calculate commonality of platformed products starts by decomposing the products into smaller subsystems or components. Key decisions that are made in this step are related to what is considered a component. Is it the entire power board, for example, or each individual component on it? Or, is it partial geometry of an injection mold (such as in many power tools) or the entire resulting part? Is it the gears or the transmission or maybe the entire power train? These choices are also needed for platform optimization whether it was optimizing the variety based on modules or attributes, for example Fujita (2005).

There are some methods that allow module decisions at varying levels of granularity (Hölttä-Otto et al. 2008), but even for that, a careful decomposition is of benefit.

## 9.2 Representation of an Architecture

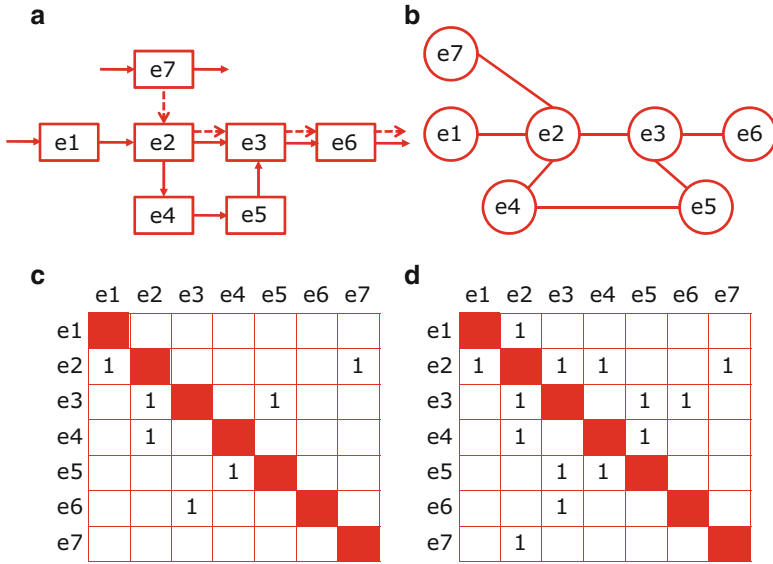
There are multiple ways of representing product architecture. Some of the common types include a functional model (Pahl and Beitz 1996; Stone et al. 2000; Otto and Wood 2001) a network model (Sosa et al. 2007), or a design structure matrix (DSM) (Steward 1981; Browning 2001) that will be briefly reviewed here.

A functional model is an abstract model of what the product does. It is a block diagram of the product's functions and their interrelationships. The functions are represented as boxes in the diagram (Fig. 9.1a). Each function interacts with the other functions via the so-called flows. These can be material, energy, or information flows. These are represented as different types of directional arrows between the boxes. For example, in Fig. 9.1a, function e1 could be importing electrical energy (solid arrow). This electrical energy is then acted upon function e2, which also receives information (dashed arrow) from function e7. Functional models are widely used in product platform development (Dahmus et al. 2001; Hölttä and Otto 2005; Hölttä-Otto et al. 2008).

Another similar way to represent an architecture is the use of network diagrams. In these diagrams the elements of the product or system form the nodes of the network. Any interaction among the elements is denoted by an arc. The arcs are often bidirectional but can also be unidirectional similar to functional models. Many authors of recent complex system works prefer the use of networks to represent the system architecture (Baldwin and Woodard 2008; Sosa et al. 2011). Another difference between the two is that only functional models incorporate the external aspects of the system in the model represented as incoming and outgoing arrows.

Both of the above representations can be converted into a matrix form. A common matrix used in architecture is a DSM. The rows and columns of the matrix can be functions, components, teams, or tasks of the system. In product





**Fig. 9.1** Examples of architectural representation (a) Functional model, (b) network, (c) asymmetrical DSM and (d) symmetrical DSM

platform development, a component matrix is often most relevant (Browning 2001). Figure 9.1c is an asymmetrical DSM illustrating a system identical to the one in Fig. 9.1a. The direction of the arrows is translated in the matrix using the following logic: Since element e1 feeds into element e2, the column e1 will have a “1” on row e2. Often any interaction, energy, material, or information is also actively received by the following element, and thus, one could think all interactions should be bidirectional. A DSM for this case is shown in Fig. 9.1d.

Any representation can be converted to the other one. In this example, there is a one-to-one mapping from functions to components, and thus, the elements in the functional model and the other three models are the same. This does not have to be the case. In this chapter the symmetrical component DSM is used. Before getting into the guidelines for architectural decomposition, we will review other past work in the area.

### 9.3 Background

The most common approach in product and system architecture development usually involves decomposition of the product, or a family of products, into subsystems. These subsystems are likely to consist of other yet smaller subsystems

as they are decomposed further. The process continues until the whole system is decomposed into small components (Pimmler and Eppinger 1994; Helmer et al. 2010; Chiriac et al. 2011a).

Some guidelines toward architectural decompositions exist. In project management, use of a work breakdown structure is common. Decomposing a project to tasks of approximately 2,500 person hours is typical (Hölttä-Otto and Magee 2006), but no clear guideline on the granularity is given. Typical rule is to reach a “manageable” level (Haugan 2002; Miller 2008). In system architecture literature, Tilstra et al. (2009) identify the problem and hint in their early work that there might be benefits to specific levels of granularity. In general, it has been assumed that there is a right way of system decomposition. Usually the process is driven by the designer’s perception and knowledge to conceptually decompose the system into a multilevel hierarchy of interacting subsystems/components. It is assumed that people familiar with the system will decompose the system in an identical manner. In reality, this is not likely. It is more likely that two different teams will have different system decompositions.

Products and systems are often decomposed following hierarchical structured decomposition. When decomposing a system, it is advantageous to decide a priori the complexity and size of the subsystems (Ariyo 2008). The multilevel hierarchical tree structure is directly affected by how “parts (small chunks/subassemblies) are classified into wholes (big chunks/assemblies)” (Ariyo 2008).

One approach to hierarchical decomposition is to use the functional basis as a guide (Otto and Wood 2001). This approach is developed for the functional model-type architectural representation and is often applied to products of low to medium complexity. Top-down decomposition of a complex system is more often done by gradually decomposing the system into chunks, major functions, or major systems. The first level of decomposition is represented by these big chunks, functions, or major systems. Each individual chunk, function, or major system is decomposed in smaller chunks/subfunctions or components/subsystems, forming the second level of decomposition and so on until the entire system is decomposed. This stage represents the last level of decomposition which is formed of individual components or functions only.

## 9.4 Choices in Decomposition

As discussed above and will be shown quantitatively later, it is not trivial to decompose a product or a system for architectural analysis and further platform development. The choices will have an effect on the rest of the platform development process. Two of the critical issues to be considered during the process are the chosen level of granularity and the decomposition viewpoint used.

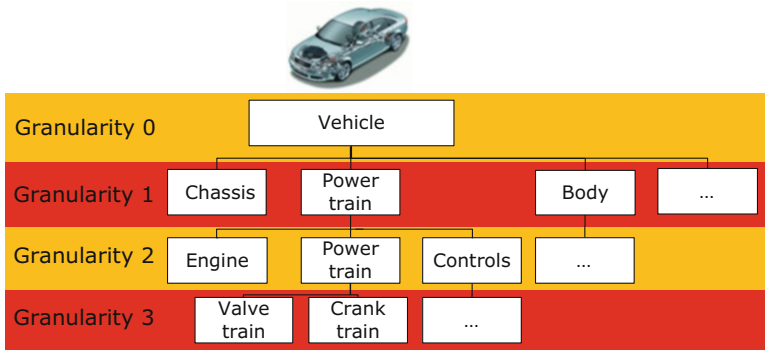


Fig. 9.2 Three levels of granularity of a vehicle

### 9.4.1 Level of Granularity

The terms decomposition, system hierarchy, and level of granularity are used throughout the systems engineering literature, but what is lacking is the analysis of how the decomposition or level of granularity affects the results and thus the conclusions of the architectural analysis. In fact, there is no literature that shows empirical or theoretical work on how to properly decompose or define the level of granularity for architectural analysis and platform development. This is especially problematic in complex system development, as opposed to simple product development, since complex systems can have multiple possible levels of granularity. The term “level of granularity,” in this chapter, is used to describe the “grain size,” i.e., the size and the detail of the system elements after system decomposition. To better illustrate the concept of granularity, Fig. 9.2 shows a partial hierarchical decomposition of a vehicle system as well as three consecutive levels of granularity. As it can be seen in Fig. 9.2, the grain size of the system elements decreases as the level of granularity increases.

Some questions arise: Which of these levels of granularity should one use for defining modules for platforms? Does it matter which level one chooses? This is explored later in this chapter.

### 9.4.2 Decomposition Viewpoint

The other issue to consider is the lens through which one looks when decomposing a product. Team or personal past experience, education, discipline, reason for modularization, and platform development all affect the viewpoint from which the architecture is decomposed. When tearing down competitors’ products for benchmarking reasons or in typical student projects, the natural approach is to

follow the disassembly of the product when recording the subsystems of the product. An equally valid approach would be to develop serviceable modules based on service expertise and thus separate controls from the mechanical parts of the system and serviceable parts from non-serviceable parts, for example. Another also equally valid approach could be to identify the main functions of the product and then decompose based on the function of each subsystem. There are surely other types of approaches, but these three are introduced here: assembly, functional, and service-based viewpoints to decomposition.

### ***9.4.3 Assembly Decomposition***

Assembly decomposition, as the name indicates, is based on the actual physical assembly, or disassembly, of the system. In practice, the system decomposition is achieved by disassembling the system virtually or by an actual teardown, depending on the scale of the system. A bill of materials (BOM) is a good starting point for this type of decomposition. Using this approach the system is first analyzed by identifying assemblies or chunks that can be easily removed from the system. The DSM for the first level of decomposition is populated by these big chunks and by mapping the connections between the chunks. If a smaller chunk seems to be attached to another, both should be kept together. Using this approach to decomposition, the removed chunks may not have an identity of their own: They can include components or chunks that perform different functions. With each big chunk that is decomposed further, a new level of decomposition is being defined.

### ***9.4.4 Functional Decomposition***

To decompose a complex system using a functional decomposition approach, the main subfunctions of the system need to be identified. The DSM for the first level of decomposition is populated by these subfunctions and by mapping the flows. Then, as these subfunctions are further decomposed into their subfunctions, another level of decomposition is revealed and the process continues until the desired level of decomposition is reached or until the system is decomposed into single components.

While not the goal, in practice the functions often follow the engineering disciplines of the company. For example, in a printing system, the so-called NOHAD (noise, ozone, heat, air, dust) functions are considered a function since those areas are covered by specific engineering specialties. Another way of grouping those same functions could be to group those with the other functions they are attached to. For example, fusing function could entail a cooling function in it, since cooling itself is not a function of the printer but a supporting function for the fusing process (that produces a lot of heat). We do not take stand which way is better, but we would like the reader to acknowledge that even within this one decomposition viewpoint, there are still alternative equally correct ways to decompose the system.

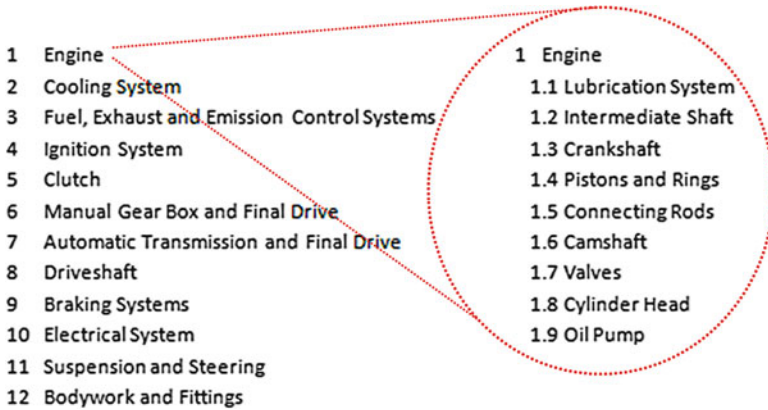


Fig. 9.3 Service manual for an Audi

### 9.4.5 Service-Based Decomposition

The service-based decomposition directly relies on the corporation’s own system description in their service manual. In the making of that manual, the system has been decomposed based on the service aspects of each system. A typical service manual consists of chapters on each high-level subsystem. All these chapters form the DSM for the first level of decomposition. Each of these chapters is further divided into subsections about the subsystems. All the subsections form the second level of decomposition DSM.

For example, there are 12 chapters in an Audi’s service manual dedicated to the high-level subsystems. These chapters represent the first level of decomposition of a system using the service-based decomposition approach. Then each of these chapters has subsections about the subsystems. The second level of decomposition is represented by the subsections.

Figure 9.3 shows the 12 chapters for the high-level subsystems on the left, and it shows the subsections for the chapter one corresponding to the engine on the right. The initial subsystems 1–12 on the left form the first level of granularity. For the second level of granularity, each of these subsystems is decomposed further—engine into nine further subsystems, for example.

## 9.5 Effect of Level of Granularity on Product Modularity

Now that we have introduced what we mean by granularity and viewpoint, it is time to show the effect they have on product platform development, specifically on degree of product modularity. Let us begin by describing the metrics we use in this work.

### 9.5.1 Metrics Used

The first metric is developed by Guo and Gershenson (2004). We call it  $M_{G\&G}$ , as shown in Eq. (9.1). It is chosen since it matches well with the definition of modularity used in this and other system architecture works. This metric measures the intra- and inter-module connectivity in a modularity matrix. The metric calculates modularity by subtracting the average inter-module connectivity from the average intra-module connectivity. In doing that it rewards for tightly coupled modules and penalizes for connections in between the modules. It is normalized in respect to the size of the system. The metric can receive values ranging from  $-1$  to  $1$ . Negative values are obtained when there is more connectivity in between the modules rather than within the modules. If all modules were fully populated (which is unlikely in real systems), the metric would vary from  $0$  to  $1$ ,  $0$  indicating an integral and  $1$  a modular system.

$$M_{G\&G} = \frac{\sum_{k=1}^M \frac{\sum_{i=n_k}^{m_k} \sum_{j=n_k}^{m_k} R_{ij}}{(m_k - n_k + 1)^2} - \sum_{k=1}^M \frac{\sum_{i=n_k}^{m_k} \left( \sum_{j=1}^{n_k-1} R_{ij} + \sum_{j=m_k+1}^N R_{ij} \right)}{(m_k - n_k + 1)(N - m_k + n_k - 1)}}{M} \quad (9.1)$$

where

$n_k$  is the index of the first component in the  $k$ th module

$m_k$  is the index of the last component in the  $k$ th module

$M$  is the total number of modules in the product

$N$  is the total number of components in the product

$R_{ij}$  is the value of the  $i$ th row and  $j$ th column element in the matrix

We also use the minimum description length (MDL) metric by Yu et al. (2005). Their metric [Eq. (9.2)] uses an information theoretic approach to calculating the coupling complexity of a matrix. In addition to providing an alternative way of addressing connectivity within and outside a module, this metric also considers potential buses in its calculation.

$$MDL = \frac{1}{3} \left( n_c \log n_n + \log n_n \sum_{i=1}^{n_c} cl_i \right) + \frac{1}{3} S_1 + \frac{1}{3} S_2 \quad (9.2)$$

where

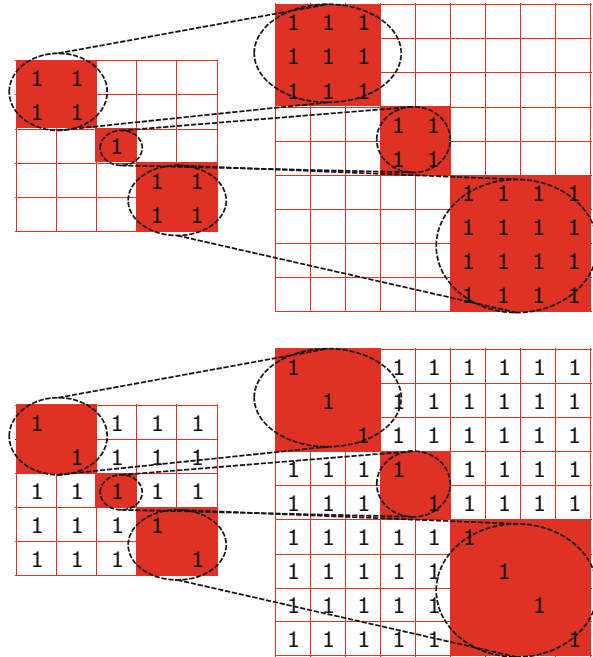
$n_c$  is the number of modules

$n_n$  is the number of rows (or columns) in the DSM

$cl_i$  is the size of module  $i$

Yu et al. (2005) provide a good tutorial on the metric, but in essence,  $S_1$  is the number of unpopulated cells that are in a module or on a bus, and  $S_2$  is the number of populated cells in between the modules and buses. The terms  $S_1$  and  $S_2$  thus describe the additional information needed to describe the DSM beyond describing

**Fig. 9.4** Fully modular (above) and integral (below) systems at two levels of granularity



simply the number and size of modules and buses. The above equation is simplified from the original version by substituting equal weights for all terms in the overall equation. A more modular matrix will result in a minimized description length. There is no upper or lower limit for this metric.

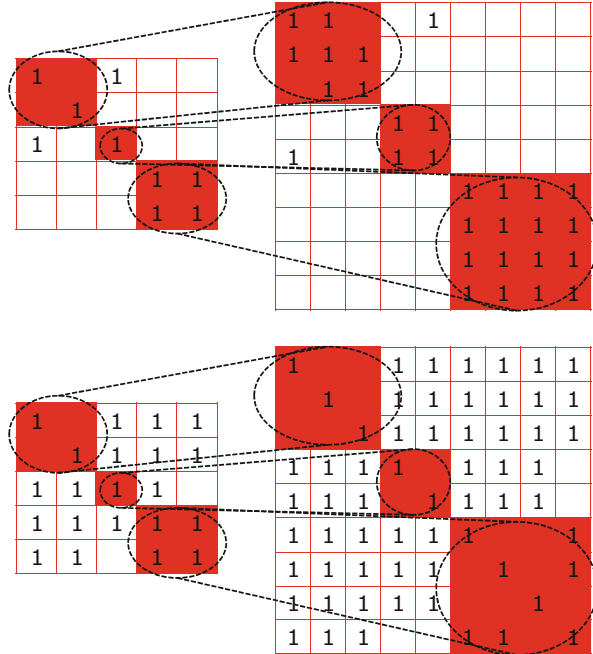
The two metrics are more independent of the coupling density than other metrics (Höltkä-Otto et al. 2012). While coupling density does affect system modularity since a denser matrix is likely more integral, it is a more interesting problem to focus on arrangement of the couplings and not the amount of coupling within the architecture. Further, calculation of simple density of the matrix will give the same results as any metric that is proportional to it.

### 9.5.2 Effect of Granularity

To show how the level of granularity affects product modularity, we generated a series of idealized matrices at two levels of granularity. A set of idealized matrices are created by varying the size of the matrices and the number of modules. The other set of idealized matrices have the location of the connections varied without changing the size and the number of modules.

Idealized DSM matrices at two levels of granularity were created as shown in Fig. 9.4. For idealized modular matrices, the modules were fully populated while

**Fig. 9.5** A slightly more integral variant of the ideal matrix (*above*) and a slightly more modular version of the ideally integral matrix (*below*)



leaving the rest of the matrix empty. The ideally integral matrices, on the other hand, were created by populating the cells outside the modules and leaving the cells inside the modules empty. This allows for moving of the connections to create alternative architectures without changing the coupling density of the matrix. We generated two variants for each idealized matrix. One sample variant for the ideally modular and ideally integral matrix from Fig. 9.4 is shown in Fig. 9.5.

A total of eight sets of four matrices were created: four sets of ideally modular and its three slightly more integral variants and four sets of ideally integral and its three slightly more modular variants. The approach is explained in more detail by Chiriac et al. (2011a). Table 9.1 shows the results for all sets created.

Figures 9.6 and 9.7 show the results graphically for the set 1 of the matrices. It is immediately evident that the measured level of modularity is different at two levels of granularity for the same system. In many cases the difference in the level of modularity between the two levels of granularity is more than the difference from one variant to the other. Of course this can be problematic in development of a modular platform where measuring a desired level of change in modularity may not produce consistent results due to the differences in the overall product decomposition. Especially in case of the integral variant, the difference in the level of modularity is dwarfed by the difference in the level of modularity between the two levels of granularity.

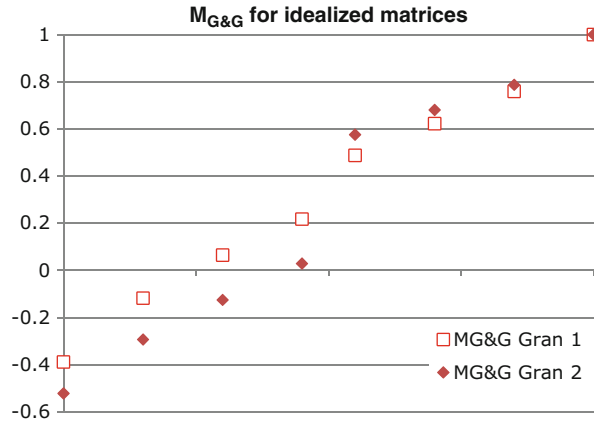
On the positive side, as can be observed in Figs. 9.6 and 9.7, the change from one variant to the other is always monotonic: Each slightly less modular variant is



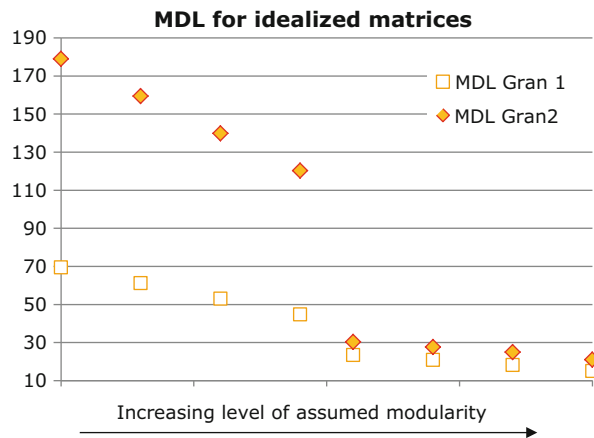
**Table 9.1** Results for the idealized matrices at two levels of granularity

Set	Metric	Granularity 1			Granularity 2				
		Ideally modular	Variant 1	Variant 2	Variant 3	Ideally modular	Variant 1	Variant 2	Variant 3
Modular variants	M <sub>G&amp;G</sub>	1.00	0.76	0.62	0.49	1.00	0.79	0.68	0.58
	MDL	15.05	18.17	20.84	23.50	20.97	24.97	27.64	30.30
	M <sub>G&amp;G</sub>	1.00	0.66	0.42	0.18	1.00	0.75	0.55	0.29
	MDL	11.00	13.67	16.33	19.00	16.14	20.14	22.81	28.14
3	M <sub>G&amp;G</sub>	1.00	0.56	0.11	-0.33	1.00	0.71	0.42	0.27
	MDL	4.00	5.33	6.67	8.00	8.42	11.09	13.76	15.09
4	M <sub>G&amp;G</sub>	1.00	0.71	0.41	0.12	1.00	0.84	0.53	0.11
	MDL	13.29	17.29	21.29	25.29	20.31	24.31	32.31	44.31
Granularity 1									
Set	Metric	Granularity 1			Granularity 2				
		Ideally integral	Variant 1	Variant 2	Variant 3	Ideally integral	Variant 1	Variant 2	Variant 3
Integral variants	M <sub>G&amp;G</sub>	-0.39	-0.12	0.06	0.22	-0.52	-0.29	-0.13	0.03
	MDL	69.454	61.23	53	44.77	179.05	159.5	139.9	120.3
	M <sub>G&amp;G</sub>	-0.61	-0.27	-0.03	0.31	-0.71	-0.46	-0.26	0.00
	MDL	141.67	123	104.3	85.67	306.5	274.8	253.7	211.5
3	M <sub>G&amp;G</sub>	-0.33	0.11	0.56	1.00	-0.71	-0.42	-0.13	0.02
	MDL	24	17.33	10.67	4	101.03	83.39	65.75	56.93
4	M <sub>G&amp;G</sub>	-0.76	-0.47	-0.17	0.12	-0.83	-0.67	-0.35	0.07
	MDL	247.03	216.5	109.4	155.3	542.93	508.5	439.6	336.2

**Fig. 9.6** Modularity of sample set 1 matrices calculated using  $M_{G\&G}$  (1 is ideally modular)



**Fig. 9.7** Modularity of sample set 1 matrices calculated using MDL (lowest number is the ideally modular matrix)



correctly recognized as a slightly less modular variant by both metrics at both levels of granularity. This is true for all sets. Similarly, the choice of which connections are changed does not alter the results (Chiriac et al. 2011a). The practical implication of this is that if the system is decomposed and analyzed at one level of granularity, the architectural change can be detected correctly if the new matrix is analyzed at the same level of granularity. The same applies for the integral variants as well when using the  $M_{G\&G}$  metric. The MDL, however, can be calculated in two different ways for the integral variants. If the mostly filled rows are considered buses, i.e., a fully integral matrix is full of buses, the value for MDL remains constant for the integral variants and is thus not useful in this particular case. However, one can also ignore the bus-like properties of an almost full matrix and calculate the MDL relative to an ideally modular matrix. These results are shown in this chapter. Ideally integral matrices constructed the way done here are not common in practice, and thus, MDL would vary from variant to other in real life no matter which way the MDL was calculated. This is also the case for the case study example as shown later in this chapter.

We conclude that the analysis of modularity at two different levels of granularity can yield different results. While the level of modularity changes with the level of granularity, the direction of change stays the same, and thus, when making changes to a system, a level of granularity is likely appropriate as long as is consistent within the same development project. This is encouraging as within any single project with the same person doing the architectural analysis, the analysis results are not sensitive to the DSM building approach.

### ***9.5.3 Effect of Viewpoint***

In addition to granularity, the other decision to be made during decomposition is the approach, or viewpoint, taken (Chiriac et al. 2011b). System decomposition is affected by many factors. For one, how experts define parts of the system as independent subsystems or modules is affected by their discipline and whether the parts are made in-house or not. For example, an engine on an airplane is typically an outsourced system. This can result in the airplane system experts to treat the engine as a lower-level module that is not decomposed further even though the engine is quite an elaborate system on its own. Similarly, mechanical engineers tend to bundle all or most of a product's electronics into a single-module "control" or function "control system," whereas the same system would be viewed very differently by electrical or software engineers.

As opposed to idealized matrices, the effect of the viewpoint is best shown via an example and case study.

### ***9.5.4 Case Study: Joint Effect of Granularity and Viewpoint***

The chosen system is a Xerox DocuColor™ 250. It is a multifunction printing system from Xerox Corporation (Fig. 9.8). It is a complex system that consists of thousands of physical parts, complex electronics, and control architecture and many lines of software code to control the printing process. The planning, design, and manufacturing of the printing system involved collaborations between several organizations within the company, including business units for assessing product feasibility in target markets, engineering unit for product architecture design and testing, and manufacturing units for efficient production of the product. In addition, collaboration with many subject matter experts from a wide variety of disciplines was required to successfully design and launch the final product due to the complexity of the printing process, paper handling, and image quality control.

For the printing system, examples of three high-level (granularity 1) subsystems include the duplex automatic document feeder (DADF—scanning and paper feeding device on top of the printing engine), the printing engine (xerographs), and paper trays (Fig. 9.8). The xerographic printing engine (Fig. 9.9 in the middle)

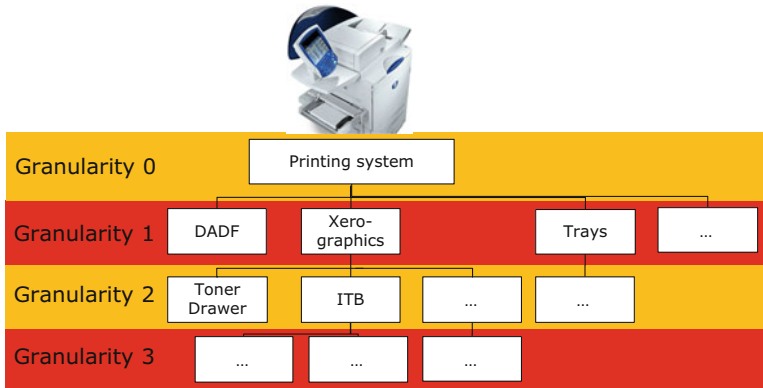


Fig. 9.8 Xerox DocuColor 250 used is the case study



Fig. 9.9 DADF module, xerographics module, and trays module

is part of the core competence of the company. It consists of many key subsystems. Each subsystem could be further decomposed into level three subsystems or even further.

The paper trays (Fig. 9.9 far right) refer to the compartments where the paper is fed into the printer. Comparing the elements at any level of granularity quickly shows that the systems are of very different levels of complexity. For example, the intermediate transfer belt (ITB) module, a subsystem whose core function is to transfer the image to the paper, is a much more complex system than the tray modules that consist only of a few plastic parts and rollers. The ITB module can be further decomposed in two more modules: the ITB belt and the ITB module drawer. Many modules/components within the ITB module drawer facilitate the image transfer. There are sensors, drives, cleaning mechanisms, and transfer rolls, just to name a few.

It is easy to see that the choice of level of granularity, or even how to decompose the system to the different levels of granularity, is not obvious, as often assumed in literature.

This printing system was decomposed from the three different viewpoints introduced earlier: assembly, functional, and service-based decomposition. In the *assembly decomposition*, an actual teardown of the product was performed. In the beginning of the teardown, the system was visually inspected to identify how covers and other identifiable chunks or subassemblies are attached to one another. Most of the components, except for the electrical and software connections, were attached either by screws or hinges. During the teardown all the connections were recorded and pictures of the printing system were taken before, during, and after a chunk was removed. The teardown started by removing all the covers and grouping those under one module named covers in the DSM. The teardown then continued on by removal of big chunks loosely attached to the rest of the product. Such big chunks constitute, for example, the paper trays, the DADF, and the xerographic drawers. All the big chunks and the covers define the DSM for the first level of decomposition. For the second level of decomposition, each chunk was further decomposed into smaller chunks. The final two architectures were approved by system experts. The two DSMs are shown below. Figure 9.10 represents the DSM for the first level of granularity from an assembly point of view. Figure 9.11 shows the matrix at the second level of granularity.

The printing system was also decomposed from the *functional* point of view. This was done in close collaboration with the system experts. For the first level of decomposition, we divided the printing system into its main functions. The main functions are electrical, controls, drives, fusing, and marking. Each of these was further decomposed to subfunctions and/or supporting functions. The resulting DSMs can be seen in Figs. 9.12 and 9.13.

Finally, for the *service-based* decomposition, the Xerox service manual for the system was followed. The first level of decomposition consists of the chapters on each high-level subsystem. The second level of decomposition consists of each subsection in each of the chapters on a high-level subsystem. The resulting DSM for the service-based decomposition is shown in Fig. 9.14 for the first level of granularity and in Fig. 9.15 for the second level of granularity. Both matrices were approved, again, by system experts.

As an example for the service-based decomposition, the first level of granularity for DocuColor 250 has 15 large subsystems or modules. After the system is decomposed into subsystems, all the connections are identified. Spatial, material, electrical, and information connections are used to populate the DSM. All the connections are identified as one in DSM regardless of their nature.

To simulate a potential architectural change in the system, a more modular matrix was created by removing the connection between the IBT module and the tray module and leaving them both connected to the control unit module rather than being connected to each other. The shaded (yellow) connections indicate the changes made to the original system in Fig. 9.14 at the first level of decomposition. In the variant, those cells are unpopulated, but they were populated in the original system.

For the second level of granularity, the major subsystems/modules were decomposed into smaller subsystems. Some of the subsystems remain the same as



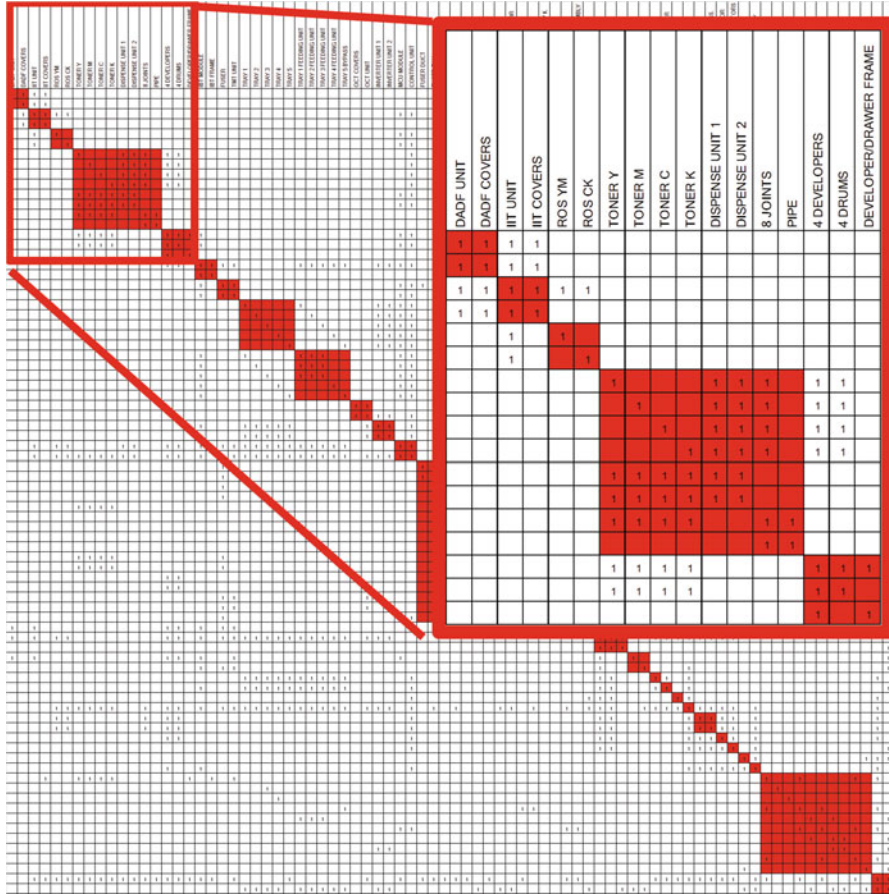


Fig. 9.11 DSM for the assembly viewpoint DSM at the second level of granularity. Close-up of the upper left corner shown in detail

Even before applying the equations, it is obvious how different each of the three pairs of matrices representing the same printing system is. The size of the DSM at the first level of granularity varies from a 15 x 15 matrix to a 38 x 38 matrix and at the second level of granularity, from a 27 x 27 matrix to an 80 x 80 matrix. Beyond the size of the matrix, there are also other significant differences between the matrices created from the different viewpoints. For example, the electrical components of the system are grouped differently in each of the viewpoints. For the functional and service-based viewpoint, at the first level of granularity, the electrical components are grouped into a 2 x 2 functional module of powering and controlling subsystems and a single service cluster (electrical/control), respectively. The same one or two modules are actually distributed around the printing system

**Fig. 9.12** DSM for the function-based architecture of the printing system at the first level of granularity

	DADF	IIT	FOS	Marking	TMT	Fusing	WASTE	TRAY 1-4	TRAY 5	OCT	INVERTER UNIT	NOHAD	DRIVES	POWER UNIT	CONTROL UNIT	CHASSIS
	1	1												1	1	
	1	1	1											1	1	1
		1	1										1	1	1	1
				1	1		1	1	1			1	1	1	1	1
				1	1	1						1	1	1	1	1
					1	1						1	1	1	1	
				1			1					1	1	1	1	1
				1				1			1		1	1	1	1
					1				1	1	1		1	1	1	1
								1	1	1	1	1	1	1	1	1
				1	1	1	1				1	1		1	1	
			1	1	1	1	1	1	1	1	1		1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1		1	1	1	1	1		1	1	1	1

such that the main drive is one module and the main control unit another, and in addition, there is a third module, another power board. The same components are grouped in three ways. The modules at the first level of granularity can be further divided into subsystems for the second level of granularity. The difference in how the electrical and control-related components are grouped into modules is very different at the second level of granularity. The 2 x 2 functional module becomes three modules similar to those identified for the assembly level at the first level of granularity, whereas the single service module is decomposed into a 11 x 11 module. The three assembly modules are also further decomposed into larger modules since all the boards consist of very distinct boards that are attached to one another.

Using Eqs. (9.1) and (9.2), we calculate the degree of modularity of the printing system and its two hypothetical variants at two levels of granularity. This was done for all DSMs of all viewpoints, and the results are shown in Fig. 9.16 in normalized form. Looking at the effect of the decomposition viewpoints, we find that using the  $M_{G\&G}$  metric, the same system is considered close to integral (functional viewpoint at granularity 1) and a quite modular (assembly viewpoint at granularity 2)



	DADF	IT	ROSYM	ROSCK	TONER DRAWER	DRUM/DEVELOPER DRAWER	IBT	TMT	FUSER	WASTE BOTTLE	JOINT LEVELER BOX ASSEMBLY	TRAY 1	TRAY 2	TRAY 3	TRAY 4	TRAY 5	OCT	INVERTER TRANSPORT 1	INVERTER TRANSPORT 2	NOHAD	MAIN DRIVE	POWER UNIT	CONTROL UNIT	UPPER CHASSIS	UPPER BACK PANEL	LOWER CHASSIS	LOWER BACK PANEL	
1	1																					1	1					
1	1	1	1																				1	1	1	1		
1		1																				1	1	1	1	1	1	
1			1																			1	1	1	1	1	1	
				1	1																		1	1	1	1	1	
				1	1	1					1												1	1	1	1	1	
				1	1	1	1				1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	
				1	1	1	1	1			1	1	1	1	1	1							1	1	1	1	1	

Fig. 9.13 DSM for the function-based architecture of the printing system at the first level of granularity

system and everything in between. The other metric, MDL, gives almost the opposite results: The printing system seems relatively modular from the functional viewpoint at the first level of granularity and mostly integral from the assembly point of view at the second level of granularity. It is obvious that a single number on modularity will not tell a full story.

While it is troubling to see the tremendous range of possible degrees of modularity for a single system, there still is a way to use modularity analysis. Table 9.2 highlights the detailed example of the service-based decomposition.

**Fig. 9.14** Variant of the printing system at the first level of decomposition from the service-based viewpoint

	DADF	IIT	Xerographics	ROS	TMT	Fuser Unit	Trays	OCT	Inverter Unit	Waste	Electrical/Control	NOHAD	Drives	Cover	Printer Chassis
	1	1									1				
	1	1		1							1			1	1
			1		1					1	1	1	1		1
		1		1							1		1		1
			1		1	1					1	1	1		1
					1	1					1	1	1		
							1				1		1	1	1
								1			1	1	1	1	1
									1		1	1	1	1	1
			1							1	1	1	1		1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
			1		1	1		1	1	1	1	1	1	1	1
			1	1	1	1	1	1	1	1	1	1	1		1
		1					1	1	1		1	1		1	1
	1	1	1	1			1	1	1	1	1	1	1	1	1

A variant was created by moving a connection between the trays and xerographic such that the information is relayed via the control unit instead (shaded/yellow cells in Figs. 9.14 and 9.15). This is an attempt to make the architecture more modular. Each of the metrics was applied to calculate the effect of this change. We find that the change in the level of modularity is far greater from one level of granularity to the other than from the original to the variant architecture. This again is of concern, but similarly as in case of the idealized matrices, we find that although there is a discrepancy in the overall degree of modularity for the system when looked at different levels of granularity, the direction of change, e.g., improved modularity (or integrality), can be detected the same regardless of the level of granularity used. According to both metrics, the hypothetical printing system where that one connection was changed is more modular than the original system.  $M_{G\&G}$  is closer to one and MDL is smaller. However, due to the large effect of the level of granularity and decomposition viewpoint, it is clear that decision regarding both needs to be made carefully during the decomposition.

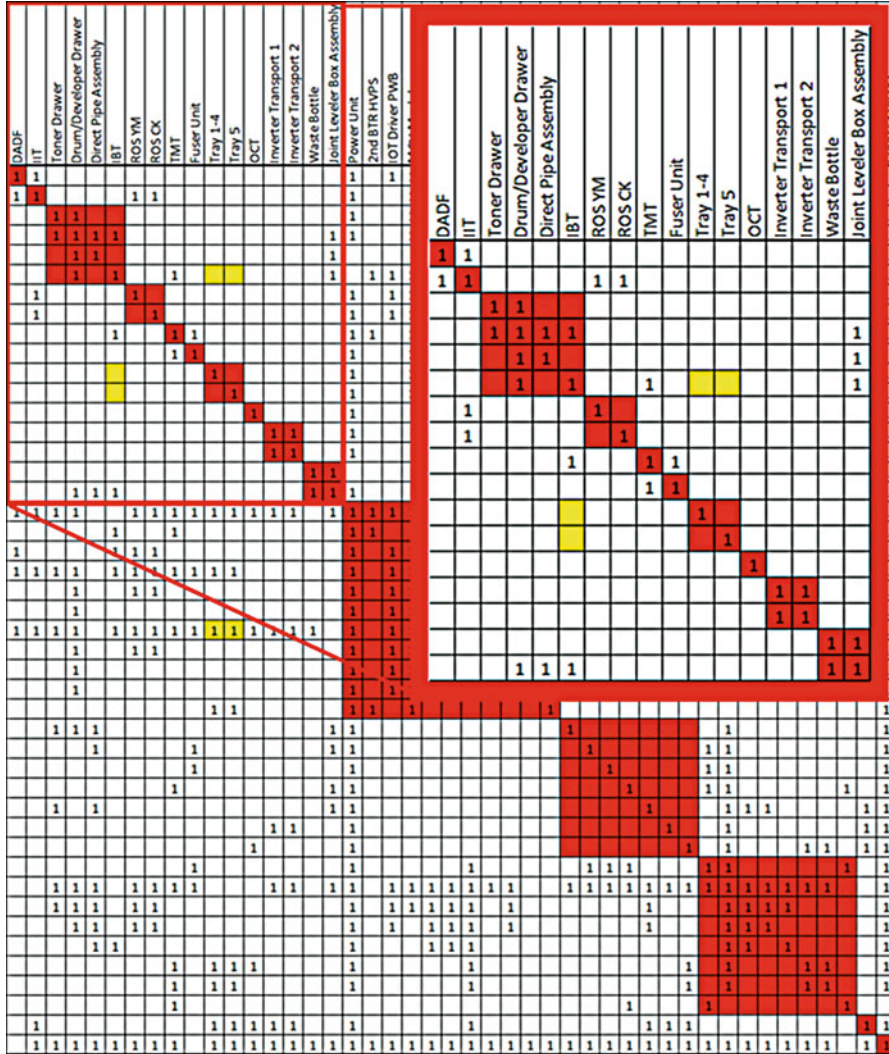
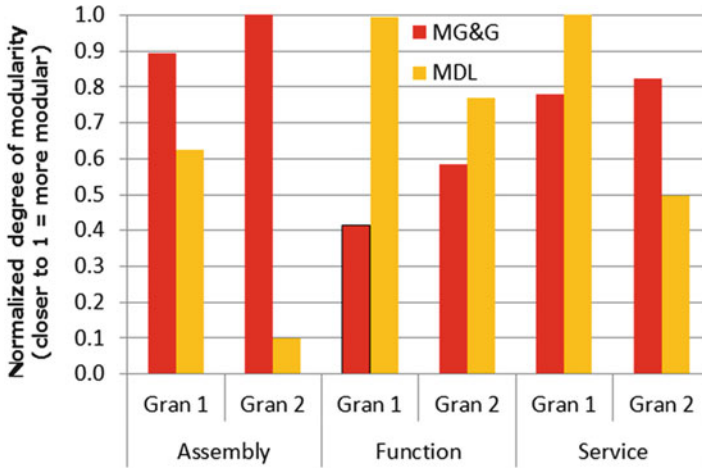


Fig. 9.15 Variant of the printing system at the second level of decomposition from the service-based viewpoint. Close-up of the upper left corner shown in detail

### 9.6 Conclusions and Recommendations

We have demonstrated how the choices on the viewpoint and level of granularity during architectural decomposition impact the resulting product architectural representation and thus the architectural analysis, specifically degree of modularity. We show that the difference from one viewpoint to the other or from a level of granularity to the other is greater than the effect of an architectural change that is



**Fig. 9.16** Level of modularity of the Xerox printing system for each viewpoint and at two levels of granularity (metrics are normalized and MDL inverted such that for both metrics closer to 1 means more modular)

**Table 9.2** Modularity of the original DocuColor 250 and its variant from the service-based decomposition viewpoint

	Granularity 1		Granularity 2	
	Original	Variant	Original	Variant
$M_{G\&G}$	0.51	0.52	0.54	0.55
MDL	456	415	2730	2722

being measured. While this is of concern, we also find that any architectural change is detected the same by all viewpoints and at either level of granularity. This results in our recommendation to calculate the change in degree of modularity rather than an absolute value of modularity. It is important to record well how the decomposition was done, from what viewpoint, and to what level of granularity, ideally including details on how all decomposition decisions are made. This is especially important in product family development, since a product platform often supports multiple products and over a long period of time. It is, therefore, unlikely that the person performing the architectural decompositions is always the same or remembers exactly what he or she did the last time.

## References

Ariyo OO (2008) Hierarchical decompositions for complex product representation. In: International design conference, Cavtat, Croatia  
 Baldwin CY, Woodard CJ (2008) The architecture of platforms: a unified view. Retrieved from, Available at SSRN: <http://ssrn.com/abstract=1265155>

- Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Trans Eng Manag* 48:292–306
- Chiriac N, Hölttä-Otto K, Suh E, Lysy D (2011a) Level of modularity at different levels of system granularity. *ASME J Mech Des* 133:101007
- Chiriac N, Hölttä-Otto K, Suh E, Lysy D (2011b) Three approaches to complex system decomposition. In: *Proceedings of the 13th international dependency and structure modelling conference*, Cambridge, MA
- Dahmus JB, Gonzalez-Zugasti JP, Otto KN (2001) Modular product architecture. *Des Stud* 22(5): 409–424
- Fellini R, Kokkolaras M, Papalambros P (2006) Commonality decisions in product family design. In: Simpson T, Siddique Z, Jiao J (eds) *Product platform and product family design: methods and applications* (1st edn. 2005. Corr. 2nd printing ed.). Springer, New York
- Fujita K (2005) Product variety optimization. In: Simpson T, Siddique Z, Jiao J (eds) *Product platform and product family design: methods an applications*. Springer, New York, NY
- Guo F, Gershenson JK (2004) A comparison of modular product design methods on improvement and iteration. In: *ASME design engineering technical conferences*, Salt Lake City, UT
- Haugan GT (2002) *Project planning and scheduling*. Management Concepts, Vienna, VA
- Helmer R, Yassine A, Meier C (2010) Systematic module and interface definition using component design structure matrix. *J Eng Des* 21:647–675
- Hölttä K, Otto K (2005) Incorporating design effort complexity measures in product architectural design and assessment. *Des Stud* 26:445–564
- Hölttä-Otto K, Magee CL (2006) Estimating factors affecting project task size in product development-an empirical study. *IEEE Trans Eng Manag* 53:86–94
- Hölttä-Otto K, Otto K (2005) Platform concept evaluation. In: Simpson T, Siddique Z, Jiao J (eds) *Product platform and product family design*. Springer, New York
- Hölttä-Otto K, Tang V, Otto K (2008) Analyzing module commonality for platform design using dendrograms. *Res Eng Des* 19:127–141
- Hölttä-Otto K, Chiriac N, Suh ES, Lysy D (2012) Comparative analysis of coupling modularity metrics. *J Eng Des* 23:10–11
- Miller DP (2008) *Building a project work breakdown structure: visualizing objectives, deliverables, activities, and schedules*. CRC, Boca Raton, FL
- Otto K, Wood K (2001) *Product design: techniques in reverse engineering, systematic design, and new product development*. Prentice-Hall, New York NY
- Pahl G, Beitz W (1996) *Engineering design: a systematic approach*. Springer, New York, NY
- Pimpler T, Eppinger S (1994) Integration analysis of product decomposition. In: *Minneapolis: ASME design engineering technical conferences-6th international conference on design methodology*
- Sosa M, Eppinger SD, Rowles CM (2007) A network approach to define modularity of components in complex products. *J Mech Des* 129:118–1129
- Sosa M, Mihm J, Browning T (2011) Degree distribution and quality in complex engineered systems. *J Mech Des* 133:101008
- Steward DT (1981) The design structure system: a method for managing the design of complex systems. *IEEE Trans Eng Manag* 28:71–74
- Stone RB, Wood KL, Crawford RH (2000) A heuristic method for identifying modules in product architectures. *Des Stud* 21:5–31
- Thevenot HJ, Simpson TW (2005) Commonality indices for assessing product families. In: Simpson T, Siddique Z, Jiao J (eds) *Product platform and product family design*. Springer, New York
- Tilstra AH, Seeparsad CC, Wood KL (2009) Analysis of product flexibility for future evolution based on design guidelines and a high-definition design structure matrix. In: *Design engineering technical conferences*, ASME, San Diego, CA
- Yu TL, Yassine A, Goldberg DE (2005) An information theoretic method for developing modular architectures using genetic algorithms. University of Illinois, Department of General Engineering. Urbana-Champaign, Illinois Genetic Algorithms Laboratory IlliGAL

# Chapter 10

## Integrated Development of Modular Product Families: A Methods Toolkit

Dieter Krause, Gregor Beckmann, Sandra Eilmus, Nicolas Gebhardt, Henry Jonas, and Robin Rettberg

**Abstract** An integrated approach for developing modular product families was developed at the PKT Institute to create individualized products for globally marketable prices. The integrated PKT-approach for developing modular product families aims to generate maximum external product variety, using the lowest possible internal process and component variety. Based on existing methods for reducing internal variety, the approach provides a toolkit of combinable method units. Tailored support is provided by this toolkit for specific needs and situations of companies facing the challenge of reducing internal variety. Several industrial case studies demonstrate how the use of one method unit or the combination of several method units supports the development of modular product families during specific corporate challenges and aims. The first section describes the challenges being addressed by the integrated PKT-approach. A survey of research fields dealing with these challenges is presented in the second section. A product family example is presented to demonstrate the state-of-the-art methods and the method units from the integrated PKT-approach. Their application in industrial projects is shown in Sect. 10.7, which is followed by the future prospects for enhancing the integrated PKT-approach.

### 10.1 Modular Product Families for Modern Market Situations

The extent that a product meets the challenges of modern market situations is determined during product development. Markets are influenced by megatrends like globalization and individualization leading to conflicting customer requirements for low prices and personalized products.

---

D. Krause (✉) • G. Beckmann • S. Eilmus • N. Gebhardt • H. Jonas • R. Rettberg  
Institute for Product Development and Mechanical Engineering Design, Hamburg University  
of Technology, Denickestraße 17, 21073 Hamburg, Germany  
e-mail: [Krause@tuhh.de](mailto:Krause@tuhh.de)

This conflict necessitates two separate product development strategies. On the one hand, the aim is to develop mass-market products to offer competitive prices through large quantities of standardized products. On the other hand, a high number of individualized products is one successful way of meeting individual customer requirements. In product development, the strategy for developing modular product families is ideal for combining advantages, such as individual customer demands, with low costs to be well prepared for the future.

The aim of developing a modular product structure for a product family is to maintain the external variety required by the market and reduce internal variety within the company. By doing this, the associated complexity of corporate processes in product development can be handled, reduced, or avoided. A major advantage of this strategy is the large number of standard modules derived that contribute to cost reduction with better utilization of economies of scale and learning curve results, especially in procurement, manufacturing, and assembly. Modular structures enable processes to be parallelized, for example, to develop different modules in parallel or to test or produce them separately.

## 10.2 Research on Reduction of Internal Variety

There is helpful support in the literature for reducing the internal variety of product families. Support primarily originates from the fields *variety-oriented product design*, *product modularization*, and *product platforms*, which are presented in this section. The basic and underlying principles of these approaches have been partly modified, adapted, or combined with new methods to form the integrated PKT-approach (Sects. 10.5 and 10.6) (Krause and Eilmus 2011).

Methods of *variety-oriented product design* focus on reducing internal variety of parts and components. An example of a methodical approach of variety-oriented product development is given by (Franke et al. 2002). They provide a framework for a variety-oriented development process, including references to several other methods that support the specific steps. The method Design for Variety by Martin and Ishii aims to derive a product platform as a robust base for future product variants (Martin and Ishii 2002; Martin 1999). Using indexes, components are identified which might be subject to change in future product modifications. Certain design principles are utilized to minimize the amount of changes necessary. Caesar and Schuh propose the Variety Mode and Effect Analysis (VMEA), which presents a cost-oriented design method for mass production (Caesar 1991). It supports the optimization of external and internal variety.

Based on these methods and further literature research, the ideal variety-oriented product structure was summarized as having four main attributes (Kipp 2012):

- Clear differentiation between standard components and variant components.
- Reduction of the variant components to the carrier of differentiating properties.
- One-to-one mapping between differentiating properties and variant components.
- Minimal degree of coupling of variant components to other components.



These attributes define the underlying principles in the method unit Design for Variety within the integrated PKT-approach.

During *product modularization* the degree of modularity has to be adapted to the corporate strategy. An adequate product modularization can yield many benefits for exploitation. Basically, a module is a group of components that exhibits stronger inner couplings than external ones. The modularity of a product can be defined as a gradual characteristic given by a set of five gradual attributes, which are commonality, combinability, function binding, interface standardization, and loose coupling of components (Salvador 2007; Bles 2011). Factors taken into account when defining the modules can be technical-functional relations or strategic aspects (Jiao et al. 2007).

An approach purely based on modularization by technical-functional relations is the Modular Design Methodology by Stone. A set of heuristics is used within this method to identify modules based on functions and their connecting flows (Stone 1997). Pimmler and Eppinger present the Integration Analysis Methodology—another approach based on technical-functional aspects—but use the Design Structure Matrix (DSM) and an algorithm that derives the optimal modularization (Pimmler and Eppinger 1994). Extending the idea of the DSM to several domains (e.g., development teams), Structural Complexity Management uses the Multiple-Domain Matrix (MDM) and provides a generic approach for analyzing and designing complex systems (Lindemann et al. 2009). An example of modularization by strategic aspects is the Modular Function Deployment (MFD, Erixon 1998), from where the module drivers of the integrated PKT-approach have been adapted. A practical application study is presented in Sect. 10.4 including the Integration Analysis Methodology and Modular Function Deployment.

An important strategy for structuring product families is the *product platform*, which can be seen as a common base of components, processes, knowledge as well as persons or relations. It provides a base for the designer to derive product variants efficiently so that faster market entries can be accomplished (Meyer and Lehnerd 1997). A higher number of standard parts can enable economies of scale at the same time (Jiao et al. 2007). Simpson proposes the Product Platform Concept Exploration Method (PPCEM)—an approach focusing on scalable product platforms (Simpson et al. 2006)—whereas configurational approaches focus on deriving variants by adding optional or individual configuration modules, such as the Platform Planning Process by Robertson and Ulrich (Robertson and Ulrich 1998).

### 10.3 Example of a Product Family

The simplified example of a family of herbicide spraying system is used to demonstrate the use of tools and methods. The MANKAR-Roll family by Mantis ULV consists of Ultra Low Volume (ULV) spraying systems that enable eco-efficient distribution of herbicides. The existing product families consist of 12 actively advertised variants as well as 24 additional variants provided on special





**Fig. 10.1** Product variants of the product family of herbicide spraying system MANKAR-Roll (Blees et al. 2010)

request (Fig.10.1, Blees et al. 2010). For example, these variants adjust the spraying systems to the individual application conditions of the customers. Different applications at in-row cultivations or public places, for example, are supported by different spray widths or sizes of wheels.

#### 10.4 Needs in the Development of Modular Product Families

The development of modular product families is still seen as a major challenge by our industrial partners; to understand why, research on needs and how they are met by existing methods was carried out. Theories in the literature were studied and supplemented by practical application of individual methods. The practical part included industrial case studies as well as workshops with industrial practitioners and engineering design students applying and comparing different methodical tools to the herbicide spraying system example.

Figure 10.2 shows how the Design Structure Matrix (Pimmler and Eppinger 1994; Eppinger and Browning 2012) is applied to the herbicide spraying system. Spatial, energetic, informatics, and material couplings between all components are allocated. The matrix is then re-sorted so that the couplings are shifted to the diagonal. This forms clusters that indicate possible modules. The DSM is a powerful tool in understanding the technical-functional conditions for a modular structure.

The Module Indication Matrix (MIM) as a tool within MFD (Erixon 1998) deals with product-strategic module drivers, describing the strategic reasons why components should be integrated into modules. Figure 10.3 shows an example of the MIM applied to the family of herbicide spraying systems. Here the components

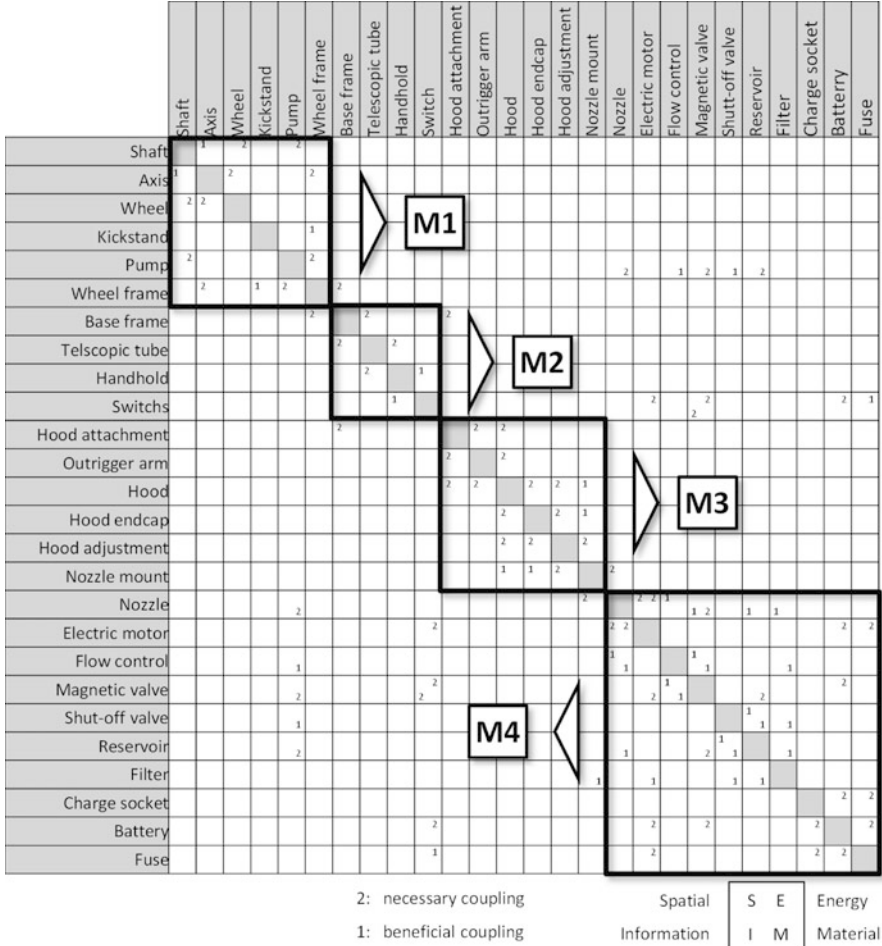


Fig. 10.2 Example of a DSM for the family of herbicide spraying systems (Krause et al. 2013)

are allocated to the module drivers that they are affected by. Modules are designated as components sharing similar module drivers or module driver patterns.

Experiences in workshops and case studies showed that method users appreciate and need the support that both tools (DSM and MIM) provide in understanding technical-functional and product-strategic module drivers. However, when working with matrix-based approaches, product-related visualizations that enable more intuitive perception are missing. The methods give no direct indication of how to reduce component variety within the product family.

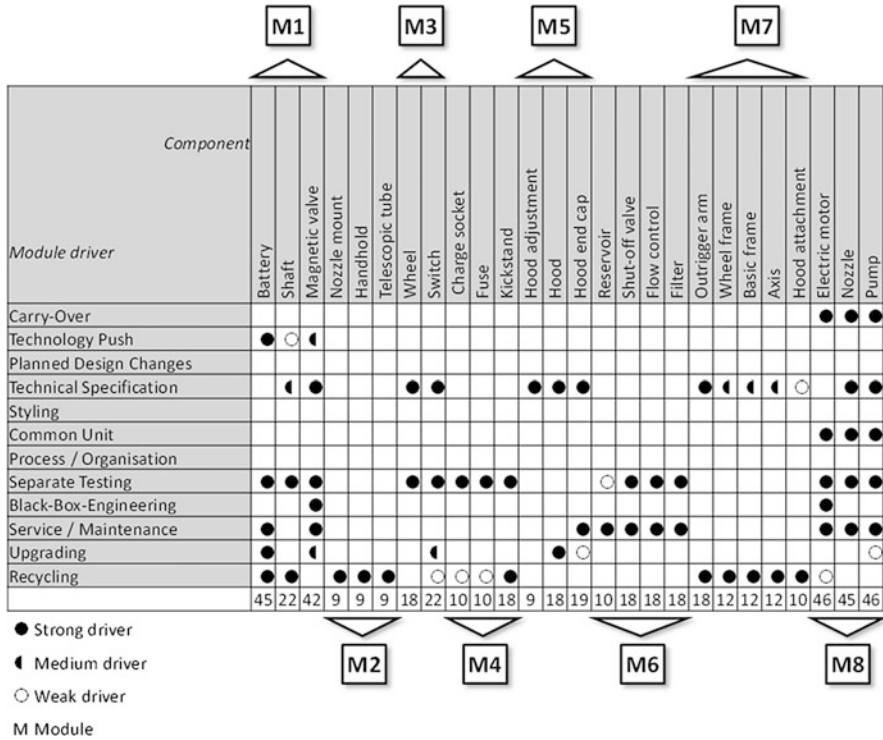


Fig. 10.3 Example of a MIM for the family of herbicide spraying systems (Krause et al. 2013)

### 10.5 Integrated PKT-Approach for Developing Modular Product Families

Based on needs in the development of modular product families, as described in Sect. 10.4, the following aims were set for the integrated PKT-approach:

- Integration of technical-functional and product-strategic approaches.
- Adaption of established ideas and tools.
- Inclusion of design for variety approaches, indicating design solutions, and reducing component variety.
- Product family-related visualizations for every design aspect.
- Fostering of team discussion.
- Flexibility of tailored support to corporate situations.

To achieve these objectives, the integrated PKT-approach offers a set of method units (Sect. 10.6) to develop modular product families. These units are structured according to their application into:

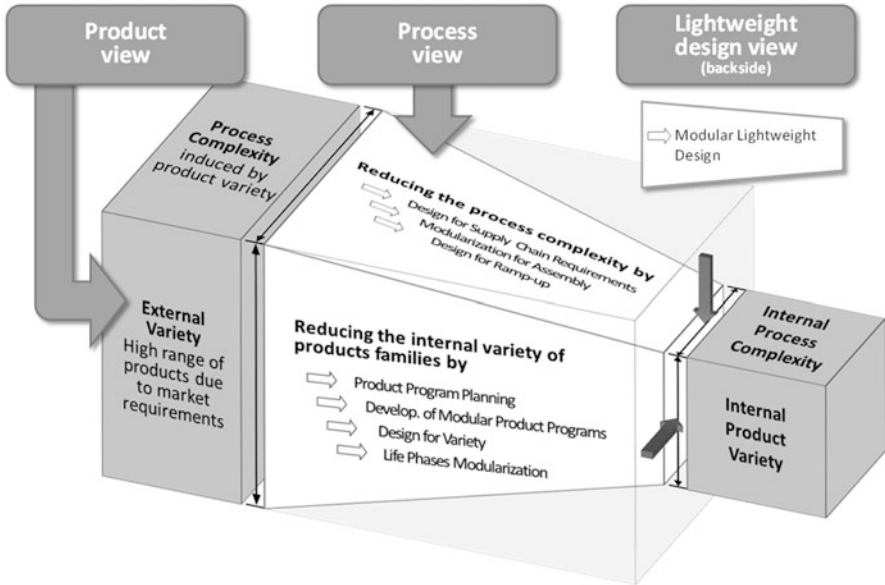


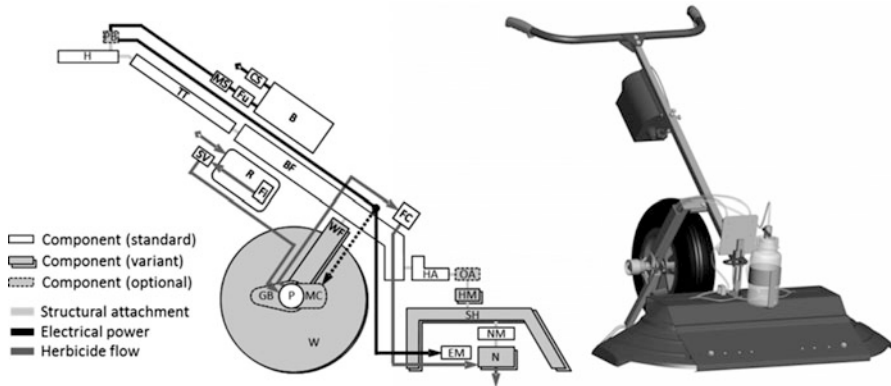
Fig. 10.4 Integrated PKT-approach incorporating three views of product family development

- The product view
- The lightweight design view
- The process view (Fig. 10.4)

The product view aims to reduce the internal product variety, while the external variety (from the customer's perspective) remains unaffected. The method units *Design for Variety* and *Life Phases Modularization* consider optimizations at the product family level, while the units *Product Program Planning* and *Development of Modular Product Programs* investigate whole product programs to achieve broader synergy effects.

Method units of the process view take the effects of product structures on corporate processes (e.g., order fulfillment and assembly) into account. To provide support for development of products driven by mass reduction needs, the integrated PKT-approach contains the lightweight design view on product family development. These method units are under development (Sect. 10.8), and this chapter focuses on the product view.

The integrated PKT-approach provides several specialized visualization tools for communication, documentation, and decision-making. The tools visualize only the information that is necessary for answering a specific question. The investigated information could be properties, functions, working principles, components, and relationships between all those. Achieving acceptance in industrial practice, user-friendly application, relevance of results, and ease of visualization in everyday engineering design practice is a high priority. For example, the developed *Module Interface Graph* (MIG, Fig. 10.5) is used to represent the spatial arrangement,



**Fig. 10.5** Optimized Module Interface Graph (MIG, *left*) of a herbicide spraying system product family (Blees 2011)

module boundaries, mark variant, or optional components and to develop module interfaces within a product family. Compared to 3D CAD data, it provides a simplified 2D view that additionally includes the flows, e.g., fluid flow or electrical power. Including the simplified spatial dimensions of the real product in MIG ensures that technical and functional aspects are continuously present during product family development. At the same time further product information are deliberately left out in order to simplify the illustration and to emphasize the relevant contents (Blees 2011). The MIG is well accepted by the industrial partners—especially compared to other product family models like function structures—and is an efficient tool for discussion between different departments (e.g., R&D and Marketing, Blees et al. 2010).

## 10.6 Method Units

Several method units use the tools of the integrated PKT-approach to fulfill specific aims (Fig. 10.6). These method units are described in this section.

Their application in industrial cases and their combination to fulfill specific corporate needs are presented in the next section.

### 10.6.1 Design for Variety

Design for Variety aims to bring the product families closer to the ideal of a variety-oriented product structure, as presented in Sect. 10.2. In the first step of the method, the external market-based and internal company varieties of the product family are analyzed. A *Tree of External Variety (TEV)* aids analysis of the external variety

	Design for Variety	Life Phase Modularization	Product Programm Planning	Development of modular Product Programs
<b>Visual tools of the PKT-approach</b>				
Tree of Variety (TEV)			✓	✓
Product Family Functional structure (PFS)	✓			✓
Module Interface Graph (MIG )	✓	✓	✓	✓
Variety Allocation Model (VAM)	✓			✓
Module Process Chart (MPC)		✓		
Program Structuring Model (PSM )			✓	
Carryover Assignment Plan (CAP)			✓	
Carryover Chart (COC)				✓

Fig. 10.6 Visual tools of the integrated PKT-approach and their application by method units

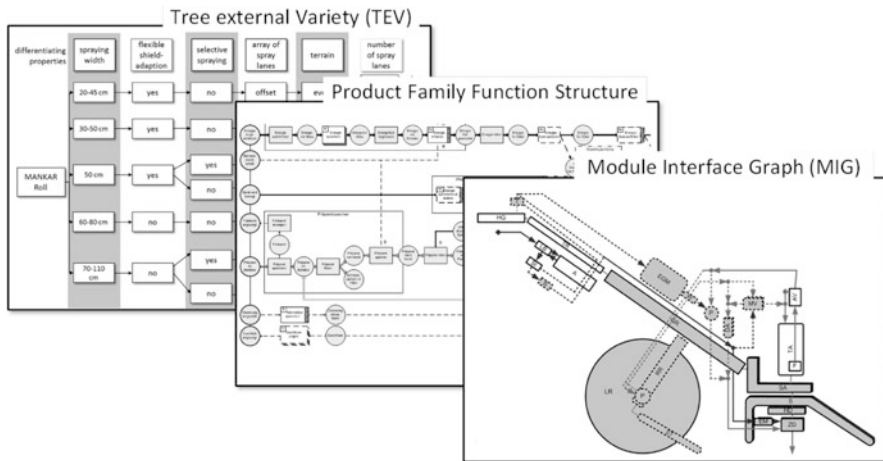
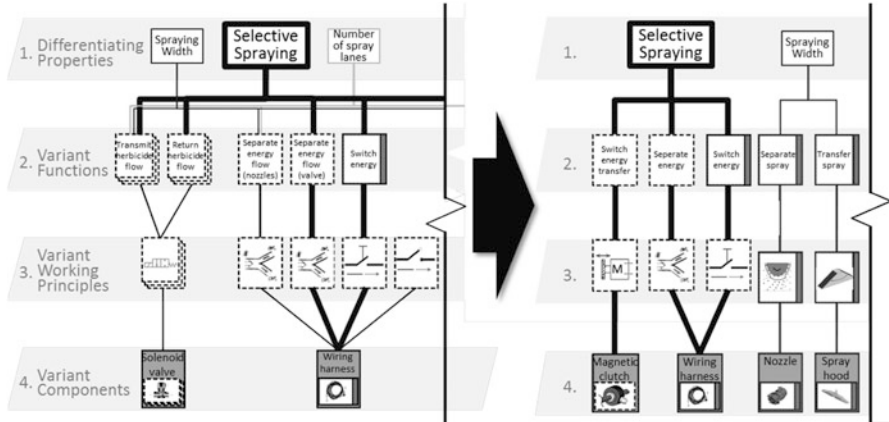


Fig. 10.7 Tools for the analysis of product variety

(Fig. 10.7). This tree visualizes the selection process of the customer by linking variant product properties relevant to customers and the offered product variants. Internal variety is analyzed at the levels of functions, working principles, and components. The variety of functions is shown in an enhanced *Product Family Functional structure (PFS)* that makes representation of variant and optional functions possible. The variety of working principles is determined from sketches, where the necessary variance of the functional elements is marked in color. The *Module Interface Graph (MIG)* is used to visualize and analyze the variety of components and connecting flows (Figs. 10.5 and 10.7).



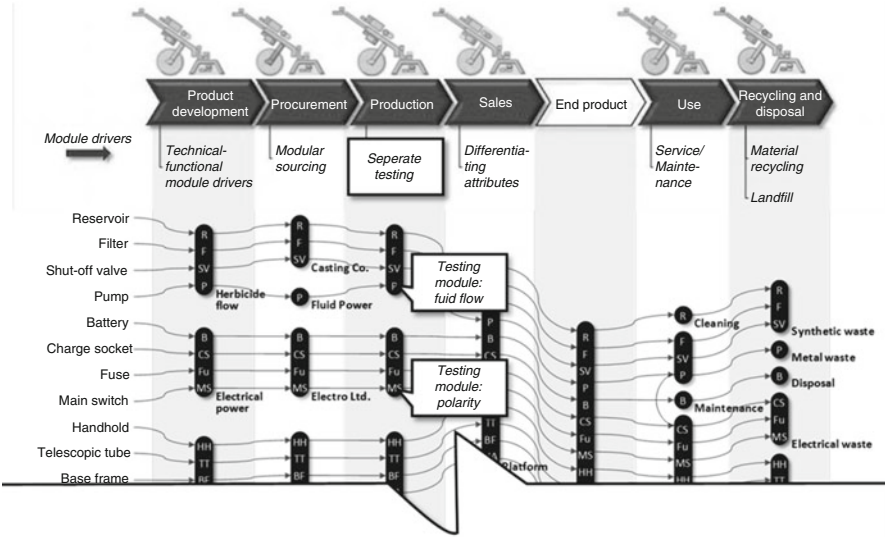
**Fig. 10.8** Applying the Variety Allocation Model (VAM) as a tool to optimize the product family of herbicide spraying systems (Kipp 2012)

All relevant information required to carry out design for variety when preparing constructive proposals is visualized in the *Variety Allocation Model (VAM)*. The connections between the levels demonstrate the allocations between differentiating properties, functions, working principles, and components (Fig. 10.8). In this way, VAM allows analysis of the degree of fulfillment of the four ideal characteristics. For variant conformity, any weak points in the design can be identified at all levels of abstraction. Thus, VAM is the basis for solution finding and selection of solutions in the methodical unit design for variety.

The result of this methodical unit is a newly designed set of components with an increased number of standard parts. Multiplication effects of the variance are avoided, with the result that each component is required in only a small number of variants. The simplified allocation structure between components and differentiating properties simplifies the variant configuration. These benefits were achieved by using the VAM as a tool to optimize product structure using a product's differentiating properties, functions, and working principles. By considering differentiating properties as well as functions and working principles, the methodical unit enriches the field of existing approaches with a method that aligns a market-oriented view with a function-oriented one.

### 10.6.2 Life Phases Modularization

Life Phases Modularization transfers the results of design for variety for each relevant product life phase to a continual module structure, while checking consistency and adjustment. Product family structure requirements can be better met by considering different product family structures for individual phases. In Life Phases



**Fig. 10.9** The Module Process Chart (MPC) as a tool for allocating module drivers and module driver specifications to modules (Blees 2011)

Modularization the life phases are considered as the phases that each produced item physically runs through. In order to emphasize the difference to the product life cycle describing introduction, growth, maturity, and decline of product generations, the term product life phases is used. The procedure is divided into the following steps:

1. Development of a technical-functional modularization
2. Development of modularizations for all relevant other product life phases
3. Combination of modularizations
4. Derivation of the modular product family structure

The starting point is the technical-functional modularization of the product development phase. Modules are provided that are largely decoupled to reduce the complexity of the development task and allow parallel development of modules. Technical-functional approaches, such as that described by Stone (Stone 1997), can be applied at this step. The development of modularization perspectives for all relevant product life phases is made by module drivers associated with individual life phases. For instance, the production phase is mapped by the module driver “Separate Testing” (Fig. 10.9).

The module drivers are a concept known from Modular Function Deployment (Sect. 10.4, Erixon 1998) that has been supplemented with concrete specifications to develop modules. In the module driver “Separate Testing,” the tests carried out demonstrate the product-specific specifications. In network diagrams, these specifications are linked to the components of the product.



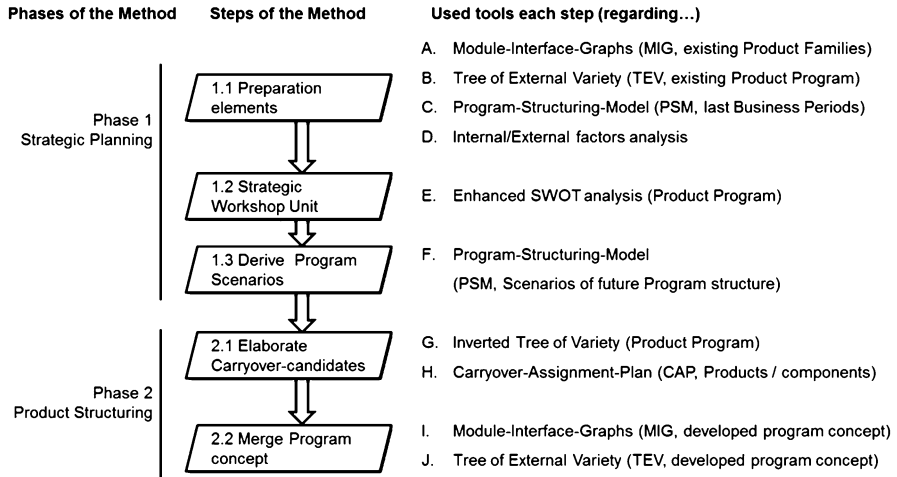


Fig. 10.10 Procedure of the method unit Product Program Planning (Jonas et al. 2012)

The preparation of modules is made by grouping the components that relate to a common module driver specification into one module. Subsequent to the development of modular product family structures for the individual life phases, the modularizations are visualized in an MIG to check consistency between life phases and find conflicts. It was found that the same module structure cannot be realized for all life phases because of the contradictory criteria. It is important that the module structures of the individual phases are adapted and continuous but not 100 % congruent. For assembly, it may be advantageous to install a module that is as large as possible. For purchase, it may be necessary to buy this module in the form of smaller modules from different suppliers which, in a well-adapted structure, must not be contradictory. The *Module Process Chart (MPC)* transparently combines the various perspectives of different life phases and makes the coordination process more clear (Fig. 10.9). Finally, the product family structure can be derived.

### 10.6.3 Product Program Planning

The method unit Product Program Planning (Jonas et al. 2012) consists of two major phases (Fig. 10.10). In the first phase, scenarios for the future composition of the product program are elaborated. The starting point is an analysis of the current condition of the program. MIGs and a TEV are used to describe the product families. For an economical and structural analysis of the program, a graphical representation called the *Program Structuring Model (PSM)*, shown in Fig. 10.12) is

used. To develop future scenarios for the program, internal and external trend factors are investigated, each considering product and stakeholder perspectives. Based on these analyses, scenarios are elaborated in a workshop unit and visualized by the PSM, step 1.3 in Fig. 10.10.

In the second phase, strategic carryover components are conceptualized for each scenario. In this step, all components are compared by their properties in order to develop concepts for carryover use. The whole program is then visualized using the *Carryover Assignment Plan (CAP)*, shown in Fig. 10.13), which contains all products and components showing their prospected carryover concepts.

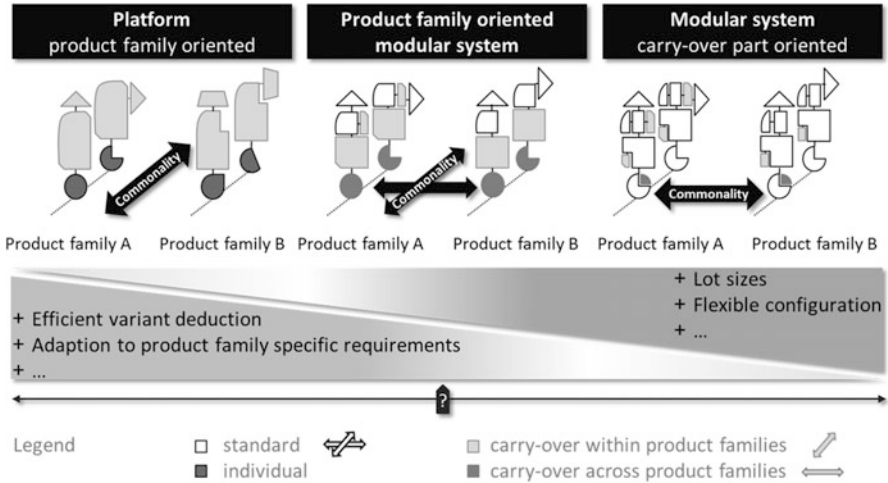
The outcomes of the method are MIGs for all conceptualized product families that show components and carryover concepts, as well as a TEV that shows the new component concepts versus market variety offered. These outcomes are used as the input for the subsequent development phases Design for Variety (Sect. 10.6.1) and Life Phases Modularization (Sect. 10.6.2).

#### ***10.6.4 Development of Modular Product Programs***

Reducing internal variety in a company can be achieved by developing modular product families but even by aligning a modular strategy across the whole product program. The aim of this method unit is to support this alignment within an existing corporate product program (Eilmus et al. 2011; Krause et al. 2013).

The methods named in Sect. 10.2, as well as the integrated PKT-approach, aim to reduce internal variety in product families at both product and process levels. Additionally, many companies expend effort on reducing internal variety over the whole product program, using various strategies for carryover of parts, components, or modules. As many market-driven factors still force differentiation, the potential for carryover often remains more at the level of standard parts rather than whole modules. In reducing internal variety, the development of module families is a solution not for standardization but for developing modules as a family of common module variants. In this context, commonality is understood not merely as the reuse of components but as any effect that makes a module seem identical to a specific system (Andreasen et al. 2004). Bringing the ideas of product family development and carryover across product families together, two major areas of action required become apparent. The first is product family oriented and deals with the development of modular product families based on modular systems or platforms to enhance commonality within a product family (displayed as diagonal arrows in Fig. 10.11). The second area of action focuses on a carryover-oriented search for modules with similar functions and customer-related differentiation properties to transfer them to a module family to increase commonality across product families (displayed as horizontal arrows in Fig. 10.11).

The extent to which these two areas of actions are relied on by a company is determined by choosing a corporate product structure strategy. This can be done by concentrating on the product family that will lead to good adaption of the



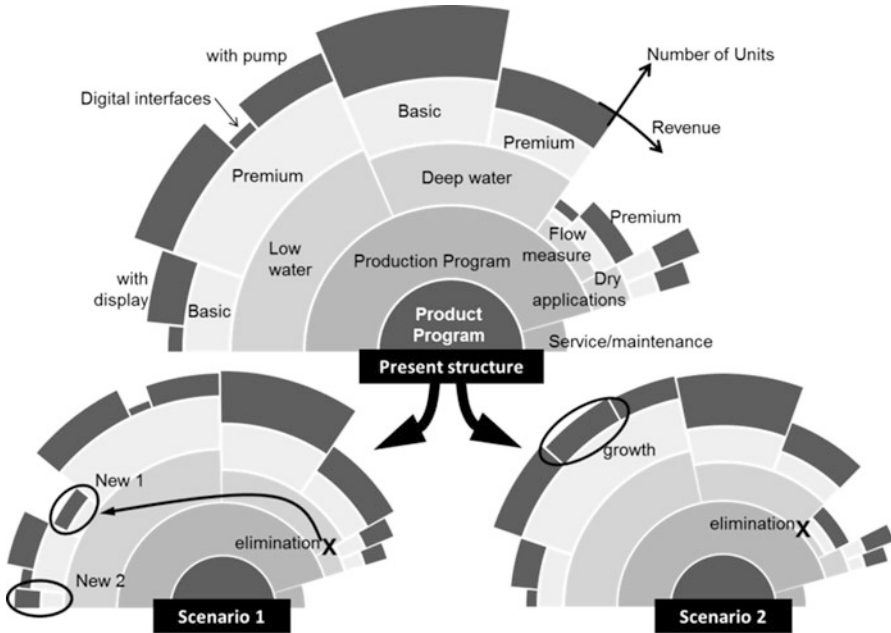
**Fig. 10.11** Product structure strategies and their focus on commonality within and across product families (Eilmus et al. 2011)

modules to the product family specific requirements (Fig. 10.11, left). This then allows efficient variant deduction, as the standard components of the product family can be designed as a platform reused in each product variant. By concentrating on a carryover-oriented view, modules are designed that enable reuse across product families (Fig. 10.11, right). This fosters high lot sizes and allows free configuration as the modules are not optimized to a specific product family. A balanced strategy is the development of a product family-oriented modular system focusing on commonality within and across product families with the same efforts (Fig. 10.11, middle).

Having defined a corporate strategy, Design for Variety and Life Phases Modularization are used to develop modular product families and module families. An example of the development of module families is shown in Sect. 10.7.3. The *Carryover Chart (CoC)* shows the potential for carryover of parts, components, and modules between product families.

## 10.7 Industrial Case Studies

The methodical tools of the integrated PKT-approach were used in several industrial case studies that combined the method units according to corporate focus and project aims. The case studies are of workshop-based projects to integrate the product knowledge, the experience, and creativity of the industrial partners involved. Four of these case studies are presented to show the application of the methodical toolkit.



**Fig. 10.12** Scenarios for water measurement devices visualized by the Program Structuring Model (PSM) (Jonas et al. 2012)

### 10.7.1 Planning a Program of Water Measurement Devices

#### 10.7.1.1 Initial Situation and Objectives

An industrial case study that demonstrates *Product Program Planning* according to Sect. 10.6.3 is presented here. The case study is modified for confidentiality reasons; the subject is an existing product program of measurement systems for water quality used in various applications ranging from the chemical industry to waste water treatment.

#### 10.7.1.2 Procedure and Application of Methods Toolkit

Figure 10.12 shows the present program structure as well as important scenarios that have been developed using step 1 of the method.

In Scenario 1, an expiring of the flow measurement systems has been prospected due to the very uncertain development of its market niche. This will lead to elimination of this unprofitable product family. It is still meaningful to offer a product that can cover this niche since it can act as an opener for system sales. Therefore, “New 1” will be aligned to the low water product line, which then has to perform a flow measurement option. It has also been identified that there is a need for a low-cost product in the low water depth segment. Therefore, “New 2” is proposed to be

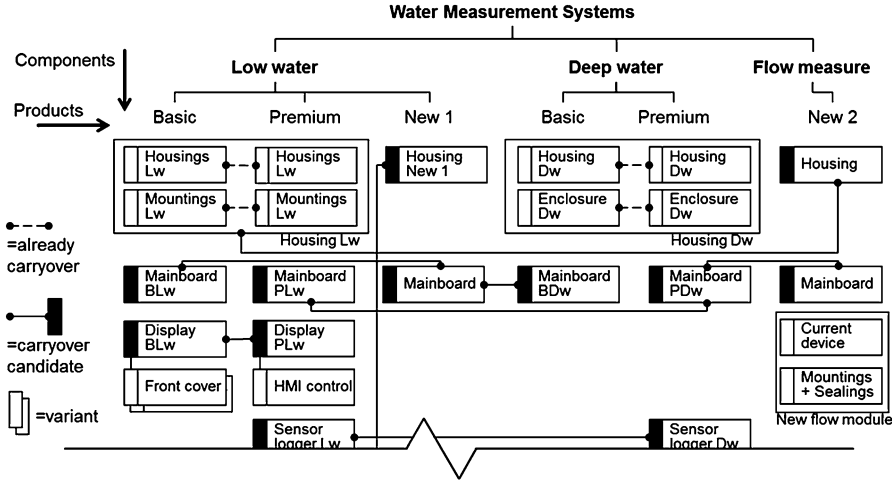


Fig. 10.13 Carryover Assignment Plan (CAP) of water measurement devices (Jonas et al. 2012)

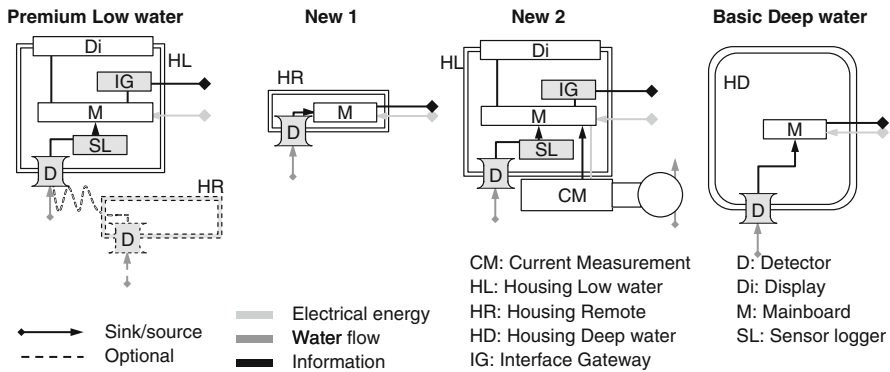


Fig. 10.14 Examples for new product concepts of water measurement devices visualized by Module Interfaces Graphs (MIG) (Jonas et al. 2012)

introduced. Development costs shall be kept to a minimum. To avoid poaching by the dry applications, it should be clearly positioned in the low depth segment.

Scenario 2 eliminates only the basic flow measurement system, as for the premium one, it still gives market potential. In this aspect, it is contradictory to Scenario 1. Regarding the low water applications, a high growth of the digital interface units is prospected. Still no standardized protocol has yet established on the market; therefore, it is not possible to predict which type of interface will grow. It is proposed to hold flexibility here in order to react to a possible technology push.

Figure 10.13 shows the common Carryover Assignment Plan (CAP) for both scenarios; therefore, it serves directly for the resulting program plan.

Selected product concepts are shown by their MIGs in Fig. 10.14. The premium low water device is equipped with the carryover mainboard, carryover display, the

decoupled interface gateway (previously realized by the additional chipset), standardized sensor logger, and the optional remote detector. “New 1” is based on the remote housing and the mainboard of the low water devices. “New 2” is based on the premium low water device and enhanced by a flow measurement module. It can serve the desired market niche but has relatively low development and production costs due to the high carryover share. The basic deep water device is now equipped with the proposed carryover mainboard.

### **10.7.1.3 Results**

Involving the different stakeholders in product planning, scenarios for the future structure of the product program have been developed and merged into a final strategy. Visualized by the Carryover Assignment Plan, a broad component share over the program has been conceptualized and forms the input for the subsequent development phases.

## ***10.7.2 Development of a Family of Gas Inlet Valves***

### **10.7.2.1 Initial Situation and Objectives**

The combined use of *Design for Variety* and *Life Phase Modularization* (Sects. 10.6.1 and 10.6.2) was accomplished in a product family development project for gas inlet valves. The valves meet the special standards for vacuum applications that had to be taken into account throughout the project. The main task was to meet the existing customer needs with one new product family while reducing the internal variety currently covered by five families of valves.

### **10.7.2.2 Procedure and Application of Methods Toolkit**

The customer needs and the current product families were analyzed in the Tree of External Variety (TEV, Fig. 10.15), which forms the basis for the first level of the VAM (Fig. 10.16). To analyze the function building in the VAM’s second level and to derive the corresponding working principles for the third level, a product family function structure was generated. The MIG is used to visualize internal component variety, building the VAM’s fourth level (Fig. 10.17, left). By analyzing the VAM, requirements for design solutions for components, working principles, and functions were identified to converge the product structure with the ideal of a variety-oriented product structure, as described in Sect. 10.2. The VAM was used for solution finding. A variety-oriented concept of a product family was derived, which was then optimized along the product life phases. This step aims to meet the requirements of a modular product structure for all product life phases and was performed with the

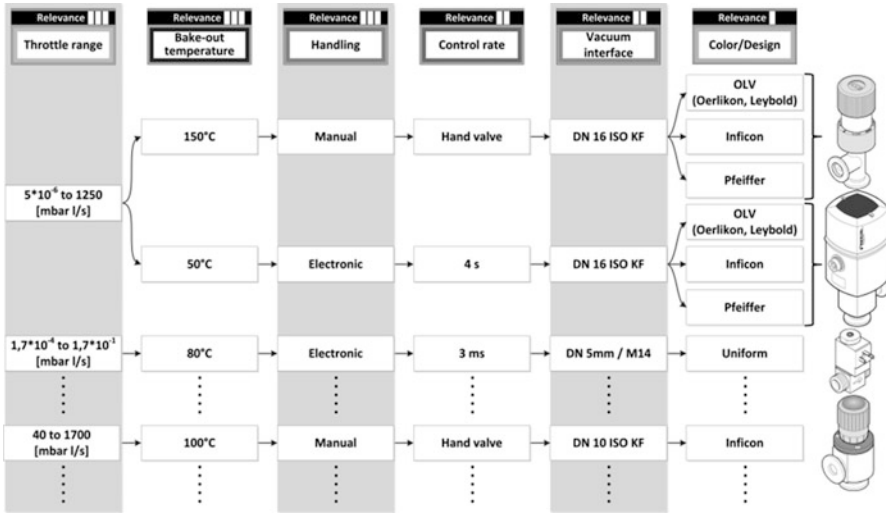


Fig. 10.15 Tree of External Variety (TEV) of the existing gas inlet valve families (Eilmus et al. 2012)

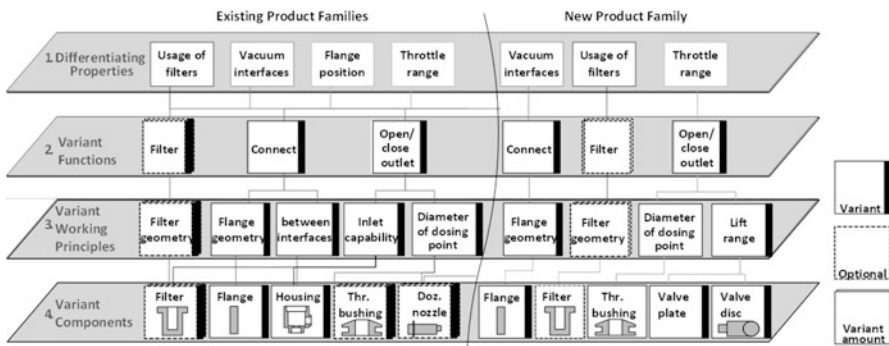
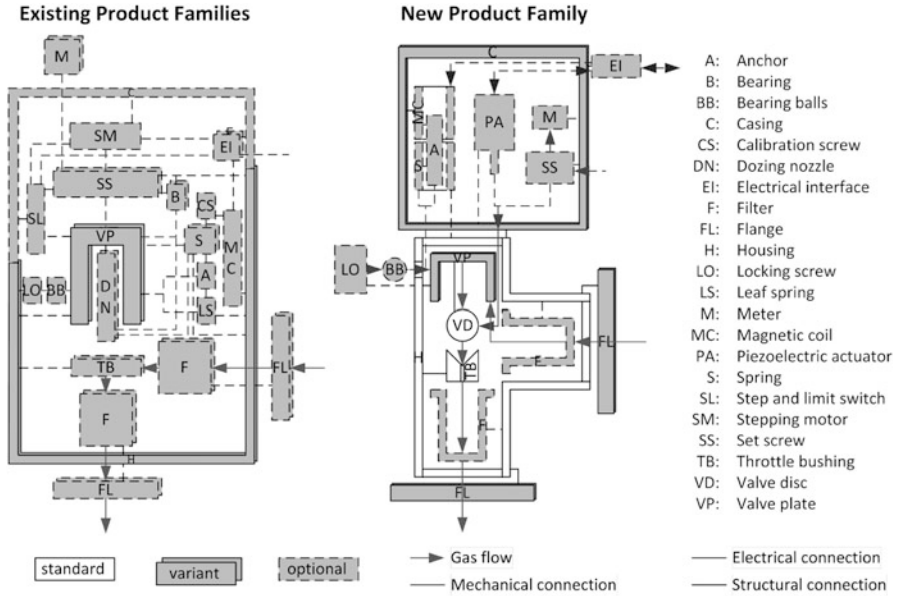


Fig. 10.16 Variety Allocation Model (VAM) of the existing gas inlet valves (left) and the new product family (right) (Eilmus et al. 2012)

help of the MPC (Fig. 10.9). For every life phase, an optimal modularization was documented, compared to the designated modularizations across the whole product life, and adjusted to derive a life phase-oriented modular concept.

### 10.7.2.3 Results

Needed components are reduced by 52 %, including the components the company has to hold in stock to build any possible variant of the family. Furthermore, the number of common components could be doubled, while the variant components



**Fig. 10.17** Module Interface Graphs (MIG) of the existing gas inlet valves (*left*) and the new product family (*right*) (Eilmus et al. 2012)

are minimized by 81 %. As these components are also physically connected, a product platform was created: all product variants can thus be configured by adding the variant and optional modules to the platform (Fig. 10.18).

### 10.7.3 A Family of Control Devices for Industrial Trucks

#### 10.7.3.1 Initial Situation and Objectives

In developing several product families of industrial trucks in separate organizational units, the production company is faced with increasing internal variety due to increasing external variety caused by global market situations. The development of module families is one way to reduce internal variety in the product program. To do this, the method unit *Development of Modular Product Programs* (Sect. 10.6.4) is applied to adapt tools and procedures of *Design for Variety* (Sect. 10.6.1) and *Life Phases Modularization* (Sect. 10.6.2) at the level of separate modules to reduce variety by configuring these modules from the same set of components. The control device is a module used in each product family of industrial trucks. Different functions are displayed in each variant according to the variety of functions of the industrial trucks, which is why 15 hardware variants of control devices were used in total. The project aim was to reduce the module variants by keeping the variety needed for proper function in each industrial truck variant.



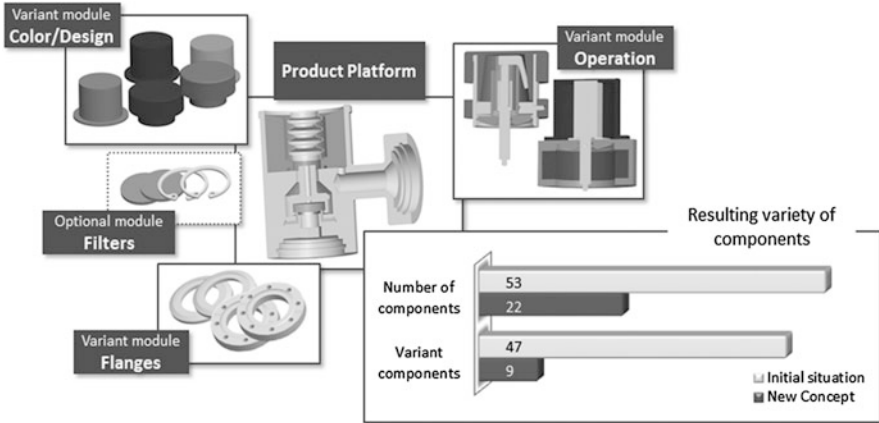


Fig. 10.18 Case study results for the gas inlet valve family

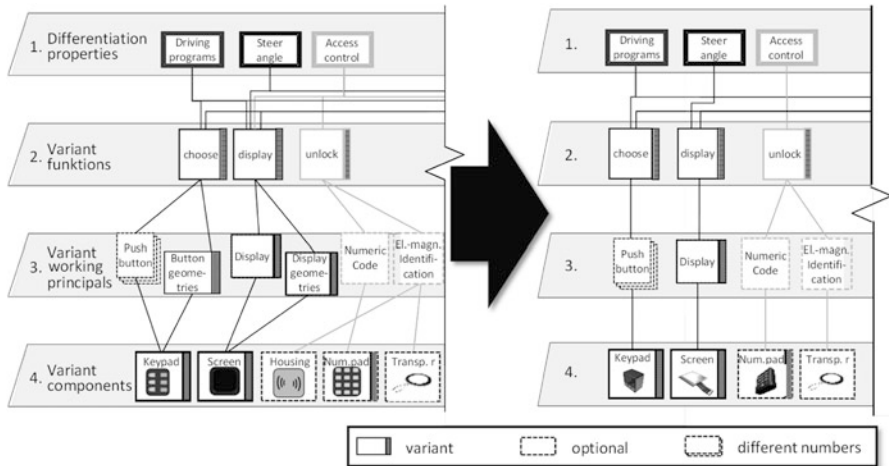
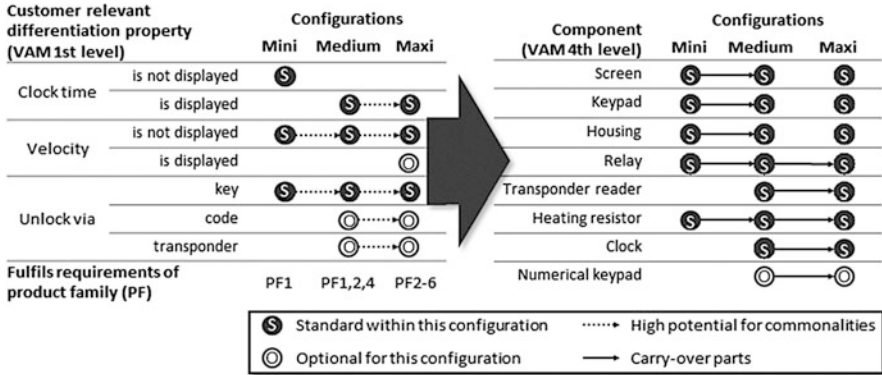


Fig. 10.19 Detail of the Variety Allocation Model (VAM) of the existing control device family (left) and the new concept (right) (Eilmus et al. 2011)

### 10.7.3.2 Procedure and Application of Methods Toolkit

As the variety of modules is strongly influenced by the variant properties required by the customers in the industrial trucks, the external variety of the control devices is much higher than in a similarly complex product that is not part of a much larger product.

In the control device, the high external variety gives little scope for standardization of hardware over all variants, as analyzed using the VAM (Fig. 10.19, right: couplings of properties between first and second level due to



**Fig. 10.20** Carryover Chart (CoC) of differentiation properties (left) and parts (right) of the control device family (Eilmus et al. 2011)

strong coupling of the devices to the properties of the industrial trucks). Because of this, a thorough analysis of potential for commonalities through component standardization among single variants is conducted using the Carryover Chart (Fig. 10.20), used as an additional tool in parallel with the VAM.

The Carryover Chart (CoC) shows the potential for commonalities (Fig. 10.20, left) and how this potential was exploited by carryover parts for some or all of the configuration variants, Mini, Medium, and Maxi (Fig. 10.20, right). These three variants were then analyzed for each life phase, collecting the conditions for common processes of the module variants or their individual components and transferring them to requirements for the embodiment design, e.g., design of common interfaces with the industrial trucks or interfaces with equipment used in the individual life phases.

### 10.7.3.3 Results

The 15 existing variant components are converted into the three modules Mini, Medium, and Maxi, which allows further optional functions when combined with the forth module: a numerical keypad. Each of the six product families uses one of the three variants as the basic control device. Some product families offer another variant as an optional high-end control device. The number of module variants was reduced by 73 % (Fig. 10.21). The modules have less than 20 components in total, a reduction by more than 75 %. The share of carryover parts, i.e., components that show no variance, increased from 1 to 29 %. The customer-required variety can be offered with less internal product variety, which means less induced process variety.

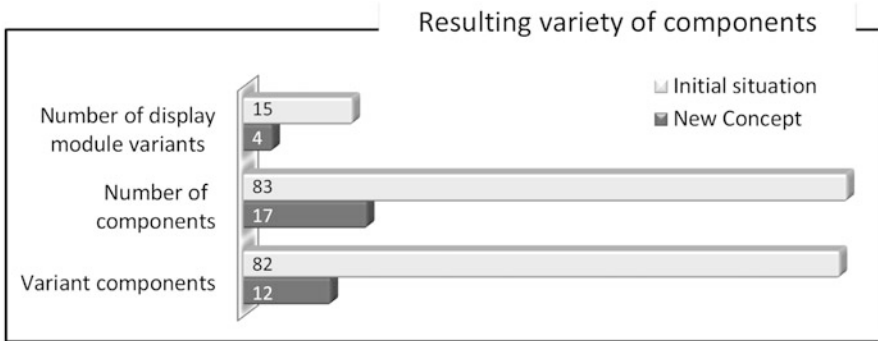


Fig. 10.21 Case study results of the control device family (Eilmus et al. 2011)

## 10.8 Perspectives on the PKT Methods Toolkit

Various completed and ongoing projects, such as the case studies described here (Sect. 10.7), have demonstrated the success of applying product family development methods. The challenge of reducing internal variety could be efficiently handled using these methods. However, case-specific new aspects of product family development have become apparent, for example, process complexity induced by the product variety in product family development. This is considered in the new method unit *Design for Supply Chain Requirements* (Brosch and Krause 2011), which is included in the process view of the integrated PKT-approach together with the units *Modularization for Assembly* (Halfmann et al. 2011) and *Design for Ramp-up* (Elstner and Krause 2011). The method unit *Modular Lightweight Design* (Gumpinger and Krause 2011) contributes to the third view of the PKT-approach. It offers the ability to judge the effects of a chosen product structure on the overall system weight of the product family, e.g., in aviation and supports the reduction in fleet weight. These methods expand the integrated PKT-approach but are not discussed in this book. Efficient adaptability of existing tools is subject of further research. Aim is to adapt and expand existing methods to fit into existing company processes and solve various rising challenges of industry. Therefore, methods and tools of the integrated PKT-approach are consolidated and collected in a methodical toolkit (Fig. 10.22). This toolkit allows case-specific combination and adaption of the evaluated methods and tools to provide answers to new challenges on common ground. Method units within the toolkit require defined interfaces to allow combinability and ensure efficient knowledge transfer. Knowledge management strategies will be used to improve communication between method units and knowledge transfer between the development team and the rest of the company. Besides, enhancement and standardization of continuous product visualization models across the units of the toolkit is subject of current research.

The future use of a consolidated toolkit is planned to start with an analysis of the initial situation and the definition of project objectives. Predefined method units could be selected and adapted according to the specific project requirements of

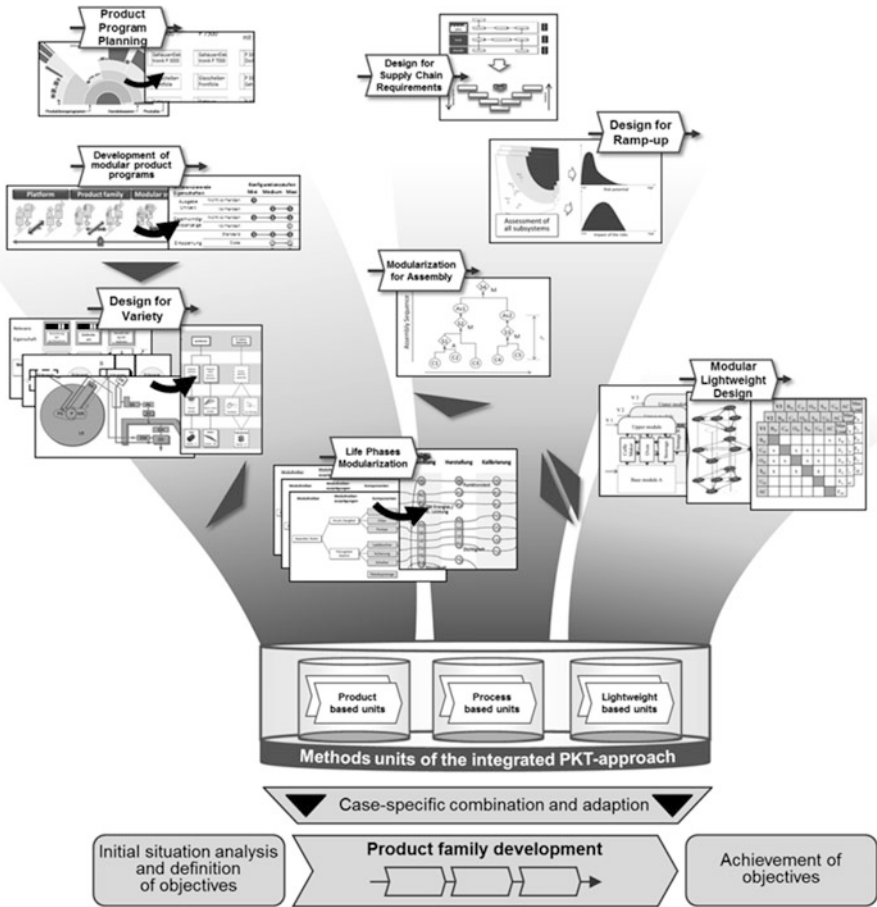


Fig. 10.22 Visualization tools and application of the integrated PKT-approach and future application of the methods toolkit

industrial partners. Serial and parallel applications should be possible to ensure problem-specific continuous support, e.g., provided by the different visualizations. After optimizing the product families, achievement of objectives can be evaluated. The broader aim of the application of a methods toolkit is to improve the applicability of existing methods and tools.

### 10.9 Conclusions

Reducing internal variety is a challenge that touches all life phases of a product family, particularly product development. The proposed methods toolkit with combinable method units provides tailored support to various corporate situations.

Therefore, branch-specific points of action can be addressed efficiently, as application in different fields of industry has shown. The branch-specific knowledge, experience, and creativity of a company's engineers are integrated using graphical tools to foster discussion and exchange of concepts. This is also supported by incorporating the method units into workshop-based procedures, focusing on interdisciplinary exchange within the company. In ten case studies, a reduction in components of about 46 % on average was achieved. Even more significant is the 75 % reduction in variant components achieved. In the coming years, further research on several toolkits will be performed to improve and enhance the integrated PKT-approach.

## References

- Andreasen MM, Mortensen NH, Harlou U (2004) Multi-product development: new models and concepts. In: Meerkamm H (ed) Design for X – Beiträge zu 21. DfX-Symposium, Erlangen
- Blees C (2011) Eine Methode zur Entwicklung modularer Produktfamilien. Dissertation, Hamburg University of Technology, TuTech Verlag Hamburg, Hamburger Schriftenreihe Produktentwicklung und Konstruktionstechnik, Band 3
- Blees C, Kipp T, Beckmann G, Krause D (2010) Development of modular product families: integration of design for variety and modularization. In: Dagman A, Söderberg R (eds) Proceedings of norddesign, Gothenburg, 2010
- Brosch M, Krause D (2011) Complexity from the perspective of the design for supply chain requirements. In: Blecker T (ed) Proceedings of the 2nd conference on the interdependencies between new product development and supply chain management (GIC-PRODESC), Mailand
- Caesar C (1991) Kostenorientierte Methodik für variantenreiche Serienprodukte – Variant Mode and Effects Analysis (VMEA). Fortschritt-Berichte VDI, Aachen
- Eilmus S, Beckmann G, Krause D (2011) Modulare Produktstrukturen methodisch in Unternehmen umsetzen – Entwicklung von Standardumfängen und Integration von Erfahrungswissen. In: Paetzold K (ed) Design for X – Beiträge zum 22. DfX-Symposium, Tutzing
- Eilmus S, Gebhardt N, Rettberg R, Krause D (2012) Evaluating a methodical approach for developing modular product families in industrial case studies. 12th international design conference – design 2012, Dubrovnik, pp 837–846
- Elstner S, Krause D (2011) Assessment of the ramp-up capability from the perspective of product development. In: Blecker T (ed) Proceedings of the 2nd conference on the interdependencies between new product development and supply chain management (GIC-PRODESC), Mailand
- Eppinger SD, Browning T (2012) Design structure matrix methods and applications. MIT Press Ltd., Cambridge
- Erixon G (1998) Modular function deployment: a method for product modularisation. Dissertation, The Royal Institute of Technology, Stockholm
- Franke HJ, Hesselbach J, Burkhard H, Firchau NL (2002) Variantenmanagement. HanserVerlag, Germany
- Gumpinger T, Krause D (2011) Development of modular product families under consideration of lightweight design. In: Culley SJ (ed) Proceedings of the 18th international conference on engineering design (ICED 11), Copenhagen
- Halfmann N, Elstner S, Krause D (2011) Product and process evaluation in the context of modularization for assembly. In: Culley SJ (ed) Proceedings of the 18th international conference on engineering design (ICED 11), Copenhagen

- Jiao J, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18:5–29
- Jonas H, Gebhardt N, Krause D (2012) Towards a strategic development of modular product programs. 12th international design conference – design 2012, Dubrovnik, pp 959–968
- Kipp T (2012) Methodische Unterstützung der variantengerechten Produktgestaltung. Dissertation, Hamburg University of Technology, TuTech Verlag Hamburg, Hamburger Schriftenreihe Produktentwicklung und Konstruktionstechnik, Band 4
- Krause D, Eilmus S (2011) Methodical support for the development of modular product families. In: Birkhofer H (ed) *The future of design methodology*, 1st edn. Springer, Berlin
- Krause D, Eilmus S, Jonas H (2013) Developing Modular Product Families with Perspectives for the Product Program. *Smart Product Engineering - 23rd CIRP Design Conference*, Springer, Bochum, pp 543–552
- Lindemann U, Maurer M, Braun T (2009) Structural complexity management: an approach for the field of product design. Springer, Berlin
- Martin M (1999) Design for variety: a methodology for developing product platform architectures. Dissertation, Stanford University
- Martin M, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. *Res Eng Des* 13(4):213–235
- Meyer M, Lehnerd AP (1997) *The power of product platform – building value and cost leadership*. Free Press, New York
- Pimmler TU, Eppinger SD (1994) Integration analysis of product decompositions. In: *Proceedings of the 6th design theory and methodology conference*, New York, pp 343–351
- Robertson D, Ulrich K (1998) Planning for product platforms. *Sloan Manag Rev* 39(4):19–31
- Salvador F (2007) Toward a product system modularity construct: literature review and reconceptualization. *IEEE Trans Eng Manag* 54(2):219–240
- Simpson TW, Siddique Z, Jiao J (2006) *Product platform and product family design*. Springer, New York
- Stone RB (1997) *Towards a theory of modular design*. Dissertation, The University of Texas

# Chapter 11

## Solving the Joint Product Platform Selection and Product Family Design Problem: An Efficient Decomposed Multiobjective Genetic Algorithm with Generalized Commonality

Aida Khajavirad, Jeremy J. Michalek, and Timothy W. Simpson

**Abstract** Product family optimization involves not only specifying the platform from which the individual product variants will be derived but also optimizing the platform design and the individual variants. Typically these steps are performed separately, but we propose an efficient decomposed multiobjective genetic algorithm to jointly determine optimal platform selection, platform design, and variant design in product family optimization. The approach addresses limitations of prior restrictive component sharing definitions by introducing a generalized two-dimensional commonality chromosome to enable sharing components among subsets of variants. To solve the resulting high-dimensional problem in a single stage efficiently, we exploit the problem structure by decomposing it into a two-level genetic algorithm, where the upper level determines the optimal platform configuration while each lower level optimizes one of the individual variants. The decomposed approach improves scalability of the all-in-one problem dramatically, providing a practical tool for optimizing families with more variants. The proposed approach is demonstrated by optimizing a family of electric motors. Results indicate that decomposition results in improved solutions under comparable computational cost, and generalized commonality produces families with increased component sharing under the same level of performance.

---

An earlier version of this chapter appeared in A. Khajavirad, J.J. Michalak, and T.W. Simpson (2009) “An Efficient Decomposed Multiobjective Genetic Algorithm for Solving the Joint Product Platform Selection and Product Family Design Problem with Generalized Commonality”, *Structural and Multidisciplinary Optimization*, 39(2):187–201 (© AIAA 2009), reprinted with permission.

A. Khajavirad • J.J. Michalek (✉)  
Carnegie Mellon University, Pittsburgh, PA, USA  
e-mail: [jmichalek@cmu.edu](mailto:jmichalek@cmu.edu)

T.W. Simpson  
Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA  
Industrial and Manufacturing Engineering, Penn State University, University Park,  
PA 16802, USA

## 11.1 Introduction

A *product family* is a group of related products (i.e., variants) that are derived from a common set of components, modules, and/or subsystems called *product platforms* to satisfy a variety of market niches. Designing a family of products is a difficult task that embodies all of the challenges of product design while adding the complexity of coordinating the design of multiple products in an effort to increase commonality across the variants without drastically compromising their individual performance (Simpson et al. 2001). This challenge manifests early in the design process wherein designers must not only specify the *platform configuration*—also referred to as *platform variable selection* or *platform selection* (Khire et al. 2006)—but also optimize the design of the platform as well as the individual variants derived from the platform.

Resolving the inherent tradeoff between platform commonality and distinct variant performance is paramount: Increasing the degree of commonality among products generally reduces total cost, but it can also compromise the ability of each variant to fully achieve the desired characteristics making it distinct and attractive to different market segments. The broad problem of product family design involves many issues, such as supply chain management, manufacturing investment, estimation of cost structures, and market positioning (de Weck 2005). We focus here on developing an improved optimization algorithm for solving the joint product family platform selection and design problem by mapping the tradeoff between increased component commonality among variants vs. achievement of distinct, exogenous performance targets for each product. The aforementioned issues could be integrated within the proposed framework through appropriate modification of objective functions and problem formulation.

In the next section, we review related optimization-based research that has sought to address this tradeoff, and in Sect. 11.3 we describe our novel multiobjective genetic algorithm (MOGA)-based approach for solving the joint product family platform selection and variant design problem with generalized commonality. In Sect. 11.4, the structure of the product family problem is exploited to decompose the all-in-one formulation into a two-level MOGA, which improves its search efficiency and scalability dramatically by reducing the search space of each sub-GA and enabling use of parallel processing. In Sect. 11.5, an example involving the design of a family of electric motors is presented and optimized, and the effects of decomposition and commonality generalization and also the complexity of the proposed method with respect to the number of variants are investigated. Closing remarks and future work are discussed in Sect. 11.6.



## 11.2 Review of Related Literature

### 11.2.1 Classification: Product Family Optimization

Numerous optimization approaches have been developed within the engineering design community during the past decade to solve the product family design problem. Simpson (2005) reviews and classifies 40 such approaches from the literature. In many of these approaches, product platforms are known or specified a priori, i.e., before performing the optimization, whereas in other instances, platform selection is determined during optimization (i.e., the platform is specified a posteriori) In a similar manner, Fujita (2002) classified product family optimization problems into three classes (see Fig. 11.1): In *Class I* problems (boxes 1 and 2), product attributes are optimized under a fixed platform configuration (i.e., the platform is known a priori); *Class II* problems (boxes 3 and 4) find the optimal module selection from predefined sets of modules (i.e., the design of each module is known a priori); and finally, in *Class III* problems (boxes 5 and 6) the product attributes and platform configuration are optimized simultaneously. It is this Class III, a posteriori problem that we refer to as the *joint product family platform selection and design problem* (or the *joint problem* for short) in that it involves determining the optimal combination of (1) *platform variable selection*, (2) *platform design*, and (3) *variant design*. Each of these decisions is generally dependent on the others (it is typically not possible to know the optimal platform without first knowing the variant design, and vice versa); so Classes I and II problems cannot generally offer optimality with respect to the full problem. Thus, we focus our attention on approaches to address the Class III joint problem.

In Fig. 11.1, the classification of methods is further refined by adding the dimension of *restricted vs. generalized commonality*: Methods that employ

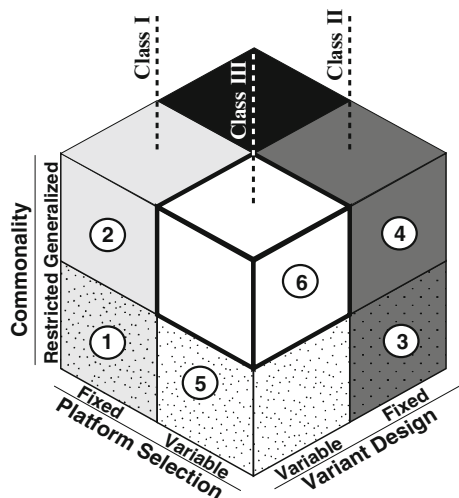


Fig. 11.1 Classification of product family optimization formulations

*restricted commonality* limit the commonality definition to all-or-none component sharing; that is, a component can either be common within the entire family or be distinct among all variants. A *generalized commonality* formulation avoids this restriction and allows for component sharing within subsets of the variants. The restricted definition is a simplifying assumption that is typically employed to decrease computational complexity; however, it imposes significant limitations that are often not observed in product family design practice (Thevenot and Simpson 2006). Therefore, there is a need for an approach capable of solving the joint problem using generalized commonality (box 6) for practical product family applications with a reasonable computational cost.

### ***11.2.2 Prior Approaches to Solving the Joint Problem***

Most of the previous a posteriori optimization methods reviewed by Simpson (2005) avoid the high computational cost of the joint problem by dividing it into multiple stages; that is, instead of addressing a Class III problem directly, in the first stage the optimal platform configuration is found followed by a Class I problem to find the variant design using the fixed platform found in the first stage. However, since platform selection and variant design are not independent, the two-stage approaches have been shown to lead to suboptimal solutions (Messac et al. 2002); therefore, single-stage approaches are preferred for optimality.

Single-stage Class III problems typically employ commonality restrictions to reduce computational cost (box 5) (Simpson and D'Souza 2004; Hassan et al. 2004; Khire et al. 2006) and, therefore, suffer from suboptimality due to the unrealistic simplifying assumptions. Fujita and Yoshida (2001) addressed generalized commonality for the joint problem (box 6) by hybridizing GA, branch and bound, and sequential quadratic programming (SQP) to determine platform configuration, direction of similarities, and variant design, respectively. The approach may be well suited to convex problems; however, for non-convex problems, it may generate suboptimal solutions due to local search in the fitness evaluation. Moreover, the nesting of optimization algorithms leads to high computational costs. Khajavirad and Michalek (2007) proposed a decomposed approach that relaxes the combinatorial platform selection variables to the continuous space and solves the generalized joint problem (box 6) through a sequence of relaxations. While the proposed method is computationally efficient, it suffers from the same problem of suboptimality for non-convex problems due to combining a heuristic (relaxation to continuous space) and a local search optimization method.

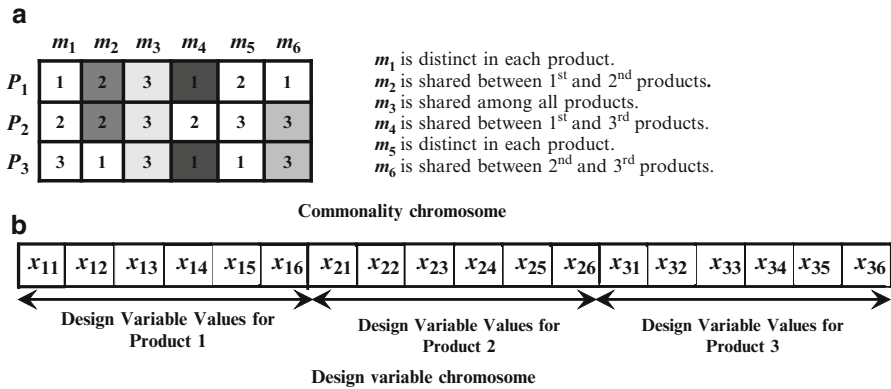
In brief, a single-stage optimization approach for solving the joint product family platform selection and design problem with generalized commonality (box 6) is needed that can solve practical problems under reasonable computational costs without employing local search methods that assume problem convexity.

### ***11.2.3 Decomposition Approaches***

An approach to reducing computational cost of the optimization problems with special structures is to decompose the all-in-one formulation into a set of interrelated subproblems such that solving and coordinating the individual subproblems is faster or more robust than optimizing the full problem all-in-one. The hierarchical structure of product families can be exploited in this way. Fujita (2002) decomposed the Class III product family optimization problem into module combination and module attribute optimization subproblems and solved subproblems in nested loops. Kokkolaras et al. (2002) applied analytical target cascading (ATC) for decomposing a Class I problem by allocating each individual product design to a separate subproblem and imposing commonality decisions by introducing subsystems with multiple parents. Michalek et al. (2006) also applied ATC to decompose a line of products including market demand and manufacturing data; however, the approach considered only manufacturing equipment sharing and did not allow for component commonality among variants. Finally, Khajavirad and Michalek (2007) introduced a two-level ATC-based decomposition scheme for solving the joint problem through a sequence of continuous relaxations; however, the approach is intended for convex formulations where design variables are continuous and gradients are available. Hence, a decomposition approach based on a non-gradient global search algorithm could avoid assumptions of convexity, continuity, or gradient availability, broadening the scope of applicability for many practical product family problems.

## **11.3 Proposed MOGA Approach**

In this chapter, we introduce a powerful new MOGA formulation for determining the Pareto front representing the tradeoff between commonality and individual variant performance in the Class III joint product family problem with generalized commonality. The underlying algorithm for our MOGA code is the elitist non-dominated sorting GA (NSGA-II) introduced by Deb et al. (2000) which has been shown to be capable of finding a well-converged and well-distributed set of Pareto optimal solutions in a reasonable computational time for many problems. However, in order to apply the original NSGA-II code to the joint problem, we have modified the chromosome representation, crossover, and mutation operators as described in the sections that follow.



**Fig. 11.2** Two parallel chromosomes for each product family in the MOGA population ( $p_i$ :  $i$ th product,  $m_j$ :  $j$ th component,  $x_{ij}$ :  $j$ th component of  $i$ th product)

### 11.3.1 Chromosome Representation

We generalize the augmented chromosome representation of Simpson and D’Souza (2004) to relax the all-or-none component sharing restriction so that platform variables can be shared among any subset of variants. This generalization is achieved by introducing two parallel chromosomes for each individual in the MOGA population (see Fig. 11.2): (1) a two-dimensional *commonality chromosome* that defines the platform configuration and allows for component sharing among subsets of products and (2) a one-dimensional *design variable chromosome* that contains design variables of all variants in the family. Hence, in a product family with  $p$  products, each defined by  $n$  components,<sup>1</sup> the commonality chromosome is a two-dimensional matrix with  $p$  rows and  $n$  columns, and the design variable chromosome contains  $np$  genes. The commonality chromosome is generated so that genes can take any integer value between 1 and  $p$ , where equal integer values indicate that the corresponding components are common.<sup>2</sup> An example of this representation for a product family with three products and six components is shown in Fig. 11.2.

<sup>1</sup> Here, without loss of generality, we assume each component is represented by a single design variable (i.e., gene); however, the algorithm is applicable to the case which each component includes different number of design variables.

<sup>2</sup> In our discussion, we assume that all products include the same number of components as candidates for commonality. However, this representation can be modified to include a general case in which any subset of components may be absent in a variant by setting the corresponding gene in the 2D chromosome to a distinct integer number (e.g., zero) and omitting those genes from the design variable chromosome of that variant. In addition, for the components that are only present in  $p' < p$  products, their commonality genes can take any integer value between 1 and  $p'$ .

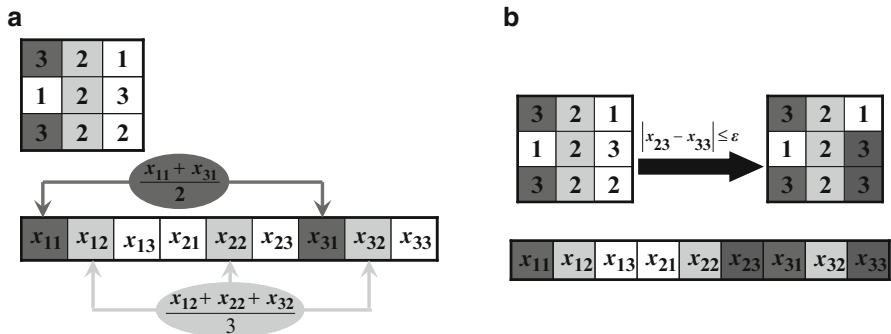


Fig. 11.3 Consistency constraints: (a) design consistency and (b) commonality consistency

### 11.3.2 Consistency Constraints

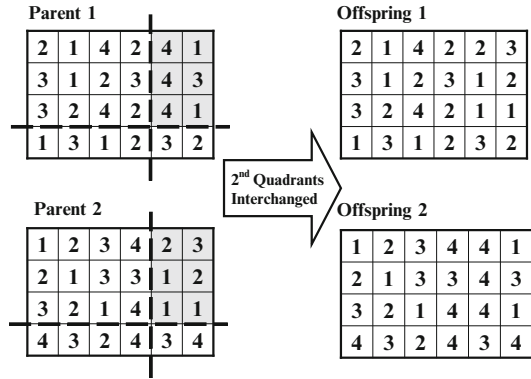
The proposed algorithm ensures that the two chromosomes remain consistent during the evolution using *consistency constraints*, which are classified into two groups: design consistency and commonality consistency. The design consistency constraints ensure that the design variables are consistent with the commonality chromosome: For each set of components identified as common by the commonality chromosome, the corresponding gene values for each individual variant are replaced by the average value<sup>3</sup> within the set (see Fig. 11.3a). The commonality consistency constraint ensures that the commonality chromosome is consistent with the design chromosomes at each iteration: if all component genes’ values for any subset of products differ by less than the maximum user-defined tolerance<sup>4</sup> as a result of the crossover or mutation operators, the corresponding genes in the commonality chromosome are modified accordingly (see Fig. 11.3b).<sup>5</sup>

<sup>3</sup> In case of discrete variables, the average value should be further rounded to the closest discrete level. Moreover, this constraint can be imposed using other strategies such as generating a random number in the upper level and sending it to the lower levels.

<sup>4</sup> The user-defined tolerance for considering two design variables to be equal should be set using knowledge about the problem, including the physical interpretation of the variable values and knowledge about the sensitivity of performance to the value of these variables. Thus, setting of the user-tolerance is necessarily case-specific.

<sup>5</sup> It should be noted that the commonality consistency constraints are only imposed for finding the optimal solution faster, and the method can identify optimal solutions without these constraints as well.

**Fig. 11.4** Two-dimensional binary crossover operator



### 11.3.3 Crossover Operators

Due to the 2D configuration of the commonality chromosome, a *two-dimensional binary crossover operator* was applied, which is a direct extension of the one-point crossover operator to two dimensions. In this operator, two random integer numbers are generated in the range of  $(1, p)$  and  $(1, n)$  to select crossover sites along  $p$  and  $n$ , where  $p$  and  $n$  again represent the number of products and components in each product, respectively. These two random numbers are used to divide the commonality chromosome into four quadrants. Then, a third random integer number, in the range of  $[1-4]$ , is generated to decide which quadrant is to be interchanged (see Fig. 11.4). The crossover type applied to the design variable chromosome is the default operator used in the original NSGA-II code, which is *simulated binary crossover* (Deb 2001).

### 11.3.4 Mutation Operators

The mutation operator is designed to mutate the platform configuration of the product family to increase the searching quality of the GA for exploring various levels of commonality. First, for each component, a random number is generated ( $0 \leq rnd_1 \leq 1$ ). If its value is less than the user-specified mutation probability ( $p_m$ ), then the corresponding component is mutated. In mutation, a new random number is generated ( $0 \leq rnd_2 \leq 1$ ). If its value is less than 0.5, then that component is set as distinct in each product and is mutated according to the *polynomial mutation operator*. Otherwise, the component is made common among all products by first being mutated in each variant and then being replaced by its average value over all products followed by a rounding strategy in case of discrete variables (see Fig. 11.5). After applying crossover and mutation operators, the algorithm modifies both chromosomes according to the consistency constraints.

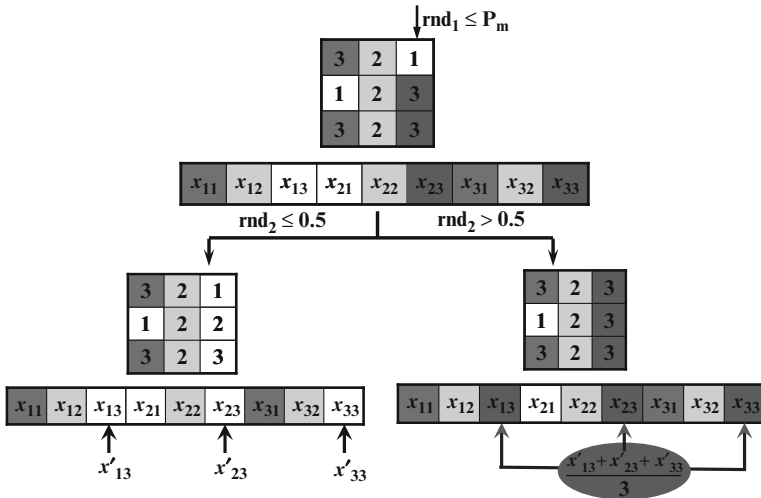


Fig. 11.5 Mutation operators ( $x'_{ij} = f(x_{ij})$ , where  $f$  is polynomial mutation operator and  $x'_{ij}$  is mutated value of  $x_{ij}$ )

### 11.3.5 Commonality Objective Function

In order to have the MOGA find the optimal platform configuration, an objective function for measuring the commonality of each family of products is added to the set of performance objective functions. Several metrics for measuring the commonality degree in product families have been proposed reflecting various commonality benefits based on company’s focus and standpoint. Khajavirad and Michalek (2007) argue that the commonality index (CI), introduced by Martin and Ishii (1996), is currently the best proxy for capturing the trend of cost savings due to component commonality relative to prior metrics used in product family optimization,<sup>6</sup> and we adopt it as the commonality objective function. CI ranges between 0 and 1 and is a measure of unique parts; that is, a higher value indicates the whole product family was made with a fewer number of unique parts: for a product family with  $p$  products each with  $n$  components, CI can be found as follows:

$$CI = 1 - \frac{u - n}{n(p - 1)} \tag{11.1}$$

<sup>6</sup>To estimate the tooling cost savings more precisely, CI should be reformulated to include coefficients representing the amount of cost saving due to sharing each component; however, this extension has no effect on the optimization approach, and all coefficients are assumed to be equal in this chapter.

where  $u$  represents the total number of distinct components in the product family. By defining  $N_i$  as the number of distinct integers for the  $i$ th component in the commonality chromosome, Eq. (11.1) can be reformulated as follows:

$$CI = 1 - \frac{\sum_{i=1}^n (N_i - 1)}{n(p - 1)} \quad (11.2)$$

Using this definition, the commonality objective function can be calculated using only the commonality chromosome while the product performance-related objectives are evaluated using each design variable chromosome; this is the key feature that enables decomposition of the proposed MOGA, as discussed next.

## 11.4 Decomposition and Parallelization of the MOGA

Aforementioned modifications to the original NSGA-II code make it convenient for optimizing the joint product family problem with generalized commonality; however, this algorithm is still only practical for problems with a relatively small number of components and variants. The commonality generalization also increases this complexity, making the all-in-one algorithm inefficient in dealing with high-dimensional problems. To address this scalability limitation, we propose a decomposition of the original formulation (see Fig. 11.6). The new method involves allocating the commonality chromosome to an upper-level GA and decomposing the design variable chromosome into its product variants, where each variant is allocated to one of the lower-level sub-GAs. In addition, the consistency constraints are imposed to all subproblems.

The general structure of the proposed model is shown in Fig. 11.7. The steps of the algorithm proceed as follows:

1. *Initialization*: Initial populations are generated (randomly or by another approach) in the upper-level and lower-level GAs independently. Next, according to the consistency constraints, both commonality and design variable chromosomes send the required data<sup>7</sup> to lower- and upper-level GAs, respectively.
2. *Fitness calculation*: The commonality metric, Eq. (11.2), and individual variant performance objectives are calculated in the upper- and lower-level GAs, respectively. The upper-level GA sends the commonality metric of each population member to all lower-level GAs, which are included in the fitness function of the corresponding individual in each sub-GA in addition to the product

---

<sup>7</sup> Data exchange necessary to enforce consistency constraints is handled through Message Passing Interface (MPI) library. Details of the implementation and a copy of the code are available through the authors or at <http://www.cmu.edu/me/ddl>.



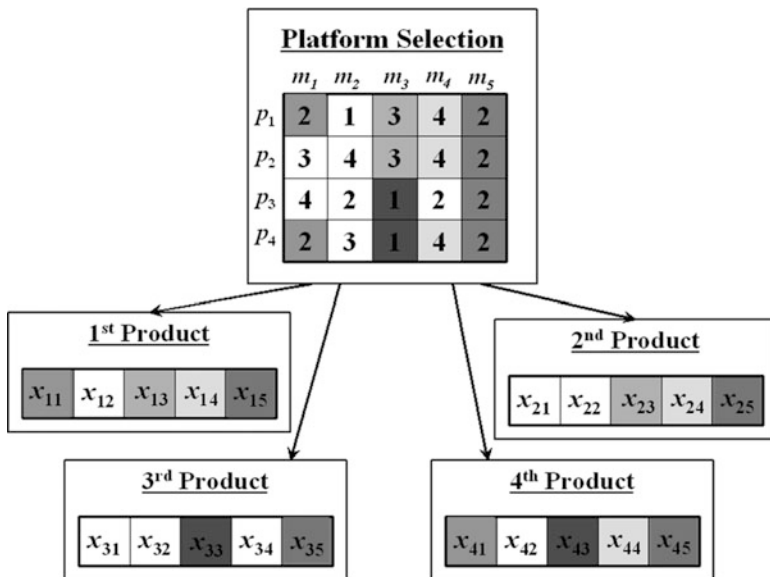


Fig. 11.6 Allocating the two parallel chromosome representations to a two-level MOGA ( $p_i$ :  $i$ th product,  $m_j$ :  $j$ th component,  $x_{ij}$ :  $j$ th component of the  $i$ th product)

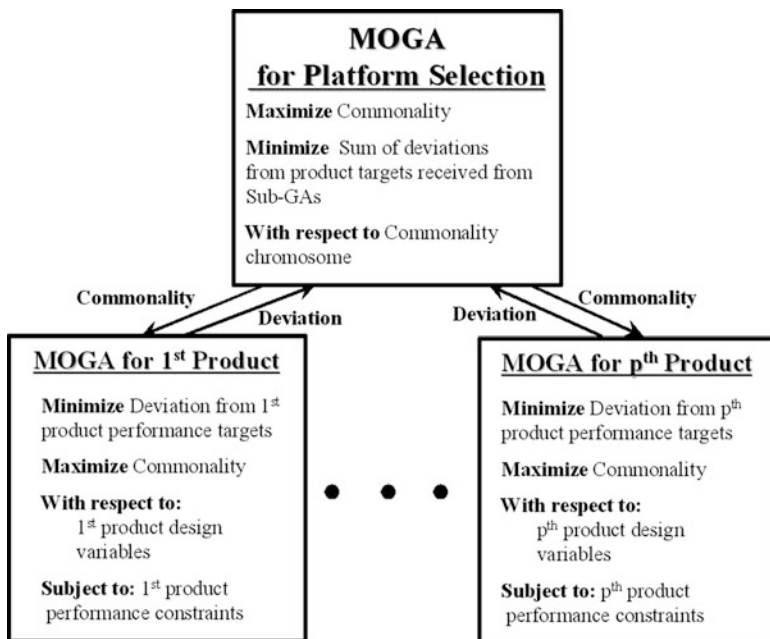


Fig. 11.7 Decomposed MOGA model for product family design

performance objectives. Each lower-level GA also returns performance deviations to the upper-level GA, which are then summed across variants to form the overall performance objective functions.

3. *Crossover and mutation*: Using the aforementioned crossover and mutation operators, offspring populations are generated in all GAs independently. Crossover and mutation operators at each subproblem are the same as the sequential version except that the tasks are divided among different processors. For example, the upper-level GA applies the two-dimensional binary crossover operator to the commonality chromosome, while lower-level GAs use simulated binary crossover. In case of mutation, the upper-level GA determines which components should be mutated, i.e., which variables should become common or distinct among the products, and passes this data to the lower-level GAs so that they can mutate the individuals accordingly. After all offspring populations are generated, they are modified according to the consistency constraints, and each lower-level GA passes back the fitness value for its corresponding performance objective(s).
4. *Replacement*: The upper-level GA combines the parent and offspring population and applies non-dominated sorting with respect to the commonality level and overall performance objectives to select the best half as the new generation that will define the new population in all (upper- and lower-level) GAs.
5. *Iteration and termination*: If the generation number is equal to the maximum generation number, then the algorithm is terminated; otherwise, go to Step 2.

Using the proposed decomposition scheme, the dimensionality of each lower-level GA remains constant regardless of the number of variants in the product family; this is the key feature making the decomposed approach scalable. However, this improved scalability is still limited by the size of commonality chromosome in that it grows linearly as number of variants increases; these desirable and adverse effects are further quantified in Sect. 11.5.5. In the all-in-one GA, selection of product families from the population is made with respect to the overall fitness value of those families; in contrast, the decomposed GA involves (1) selection of design variable chromosomes based on their sub-fitness values for producing offspring and (2) coordination of the sub-GAs after each generation to select the subset of product families from the joint parent-offspring population that will advance to the next generation. The commonality value for the entire family is also included as an additional objective function for each sub-GA which is only used for non-dominated sorting of the population prior to selection; however, since each sub-GA explores within the search space of an individual variant and selects on the basis of that sub-fitness value, the search quality enhances significantly and each lower-level GA can carry over features of high-performing subsets of the full product family chromosome to the offspring population.

While searching for high-quality individual variants in lower levels, the upper level selects the next generation members with respect to the overall family objectives by applying non-dominated sorting with respect to the commonality level and the overall performance value (formed by summing the sub-performances calculated at lower levels). Therefore, although the search for optimal platform

configuration and variant design are performed in separate GAs, the aforementioned replacement scheme applied in each generation in the upper level differentiates the proposed method from multistage methods that find the optimal platform and variant design in separate stages. Finally, due to the parallel nature of this decomposed method, each sub-GA can be executed on a separate processor using the MPI library (Pacheco 1997) for sending and receiving data among nodes during evolution.<sup>8</sup> Hence, in addition to the improved performance due to decomposition, it is possible to achieve further reductions in computational time using parallel processing.

## 11.5 Demonstration: Universal Electric Motor Family

The universal electric motor family example was first created by Simpson et al. (1999, 2001) and has since been applied as a case study by various researchers to compare the efficiency of proposed approaches with existing ones. In this example, the goal is to design a scaled-based product family of universal electric motors that satisfy a variety of torque requirements using common platforms. Hence, the optimization process involves selecting the platform and scaled design variables and their corresponding values so that the range of torque requirements is satisfied while the commonality among the motors is maximized, individual motor weight is minimized, and individual motor efficiency is maximized. The detailed analysis including all equations relating the motor design variables to the output parameters can be found in Simpson et al. (1999, 2001).

Based on this formulation, the design of a single motor involves eight design variables (see Table 11.1), two equality and four inequality constraints (see Table 11.2), treating mass minimization and power maximization as the two objectives. However, as opposed to other physical components which can be shared for reducing tooling cost, any current value (within the initial range) could be drawn from the motor based on other motor characteristics and constraints. Hence, the current is written as a function of other variables using the power equality constraint<sup>9</sup> rather than being an independent variable and therefore is not considered for component sharing. Hence, the number of design variables is reduced to 7 for each

---

<sup>8</sup> It should be noted that the parallelization method applied herein is the direct benefit of the proposed decomposition scheme and should not be confused with general types of parallel GAs (e.g., fine-grain, coarse-grain, and master-slave models), which are derived from the evolutionary nature of the GA and are independent of the specific problem being solved. Generic parallel GAs could additionally be used to solve subproblems in the proposed decomposition if the optimization of an individual product is too complex for a single GA or if further speedup is desired, but we do not pursue this possibility here.

<sup>9</sup> Power equality constraint is a second-order equation as a function of current and has two positive roots if any. Therefore, the roots are compared with respect to feasibility and objective value and the better one is picked as the current value.

**Table 11.1** Design variables and bounds for universal electric motor example

Definition	Lower bound	Upper bound
Wire turns on the armature: $N_c$	100	1,500
Wire turns on each field pole: $N_s$	1	500
Armature wire area (mm <sup>2</sup> ): $A_{wa}$	0.01	1.00
Field wire area (mm <sup>2</sup> ): $A_{wf}$	0.01	1.00
Radius of the motor (mm): $r_o$	10	100.0
Thickness of the stator (mm): $t$	0.50	100.0
Stack length of the motor (mm): $L$	1.0	100.0
Current drawn by the motor (Amp): $I$	0.1	6.0

**Table 11.2** Design constraints for universal electric motor example

1. Torque (Nm) = {0.05, 0.1, 0.125, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.5}
2. Power (W) = 300 (for all motors)
3. Feasible geometry for all motors:  $t < r_o$
4. Maximum magnetizing intensity for each motor:  $M \leq 5,000$  Amp  $\times$  turns/m
5. Maximum mass of the each motor:  $Mass \leq 2$  kg
6. Minimum efficiency of each motor:  $\eta \geq 15$  %

motor, and the power equality constraint is replaced by two inequality constraints representing the lower and upper bounds for current.

### 11.5.1 Product Family Objective Functions

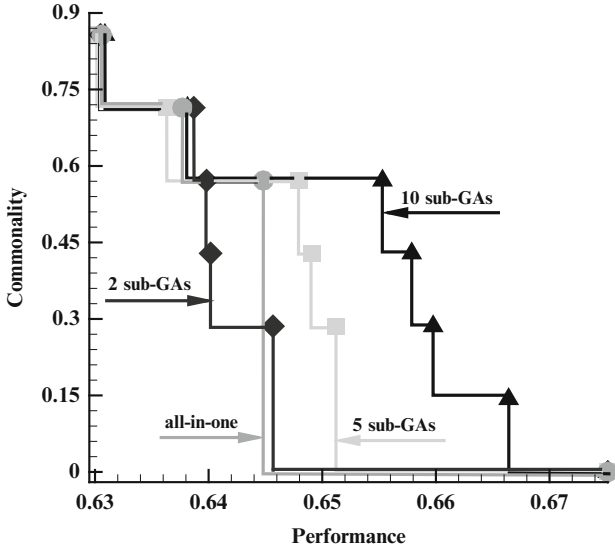
To demonstrate the tradeoff between commonality and individual performance as well as to visualize the Pareto curves conveniently, the three objectives introduced in Akundi et al. (2005) were reduced to two objectives by combining the two performance objectives into one using a fixed weighted sum:

$$f_1 = \sum_{i=1}^{10} w_1 \eta_i + w_2 (1 - m_i^*)$$

$$f_2 = CI \tag{11.3}$$

in which  $\eta_i$  and  $m_i^*$  represent efficiency and the normalized mass (mass of each motor has been normalized by dividing it over the maximum allowable:  $m_{max} = 2$  kg) for the  $i$ th motor, respectively, and  $w_1$  and  $w_2$  are the weight coefficients, which are assumed to be equal ( $w_1 = w_2 = 0.5$ ) in this chapter. Moreover, all constraints are handled using the constraint-dominated approach (Deb 2001).<sup>10</sup>

<sup>10</sup> Since GAs are generally inefficient for handling equality constraints directly and need a large population size for finding feasible solutions, the torque equality constraint has been implemented using an adaptive coefficient strategy in the constrained dominated approach.



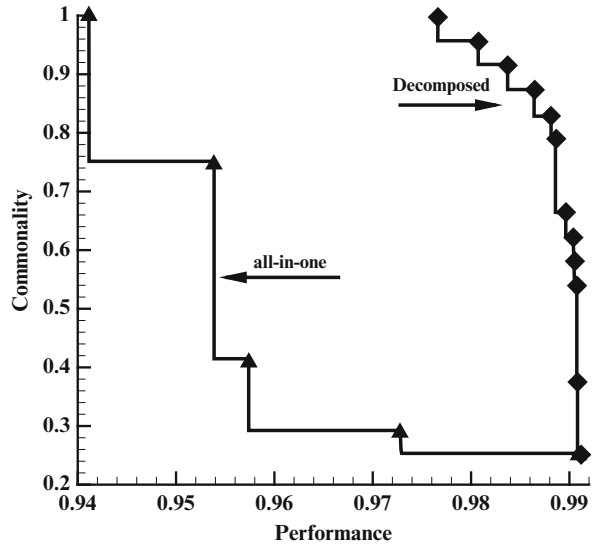
**Fig. 11.8** Pareto fronts of the electric motor family for different decomposition schemes using all-or-none commonality

### 11.5.2 Decomposition

To compare the efficiency of the decomposed approach relative to the all-in-one formulation, the electric motor family was optimized using the restricted representation for commonality genes suggested by Simpson and D'Souza (2004) using several alternative decomposition schemes. First, the product family was decomposed into ten lower-level GAs, each optimizing a single product. A population size of 2,500 and maximum generation number of 800 were used. This parameter tuning was verified by running the same code using larger values for population size and maximum generation number, resulting in negligible improvement for the Pareto curves. Next, the same problem was solved using three alternative schemes: (1) five sub-GAs, each containing two motors; (2) two sub-GAs, each with five motors; and (3) the all-in-one formulation for all ten motors. The estimated Pareto curves are plotted in Fig. 11.8. Since we are interested in finding the benefit of decomposition in producing better results for the same computational cost, the same population size and maximum generation number (which implies the same number of function evaluations) was applied in all four cases.

As can be seen from Fig. 11.8, the all-in-one formulation cannot find a well-distributed optimal front; it produces only the high-commonality portion of the curve in that this region has a lower dimensionality. By increasing the number of sub-GAs (the extent of decomposition), the Pareto curve moves toward the optimal front, and best results are obtained for the most decomposed case.

**Fig. 11.9** Pareto fronts of GAA family for decomposed and all-in-one formulations using generalized commonality

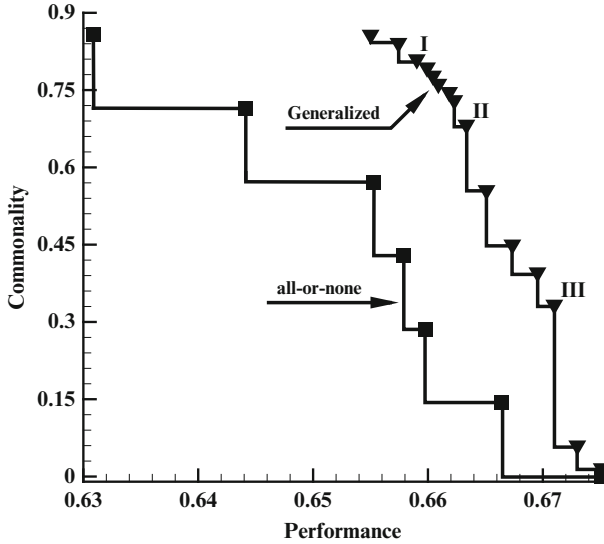


While we focus here on results from the universal electric motor example, the trends observed in this case study are also consistent with results we obtained applying the proposed decomposition to the family of three general aviation aircraft (GAA) examined in Simpson and D'Souza (2004). Specifically, the Pareto set obtained through decomposition is significantly superior to that obtained without decomposition (see Fig. 11.9) indicating the robustness of the proposed approach to solve families with different number of variants and components. Details of the GAA case study can be found in Khajavirad et al. (2007).

### 11.5.3 Generalization

As described in previous sections, using restricted all-or-none commonality definition results in a loss of the benefit of component sharing among subsets of products, which is frequently applied in practice. The scalability benefit of decomposition is even more critical for addressing this generalized case: Since adding the two-dimensional chromosome along with the design variable chromosome and searching through all possible platform configurations adds to the complexity of the algorithm dramatically, the all-in-one formulation becomes impractical even for smaller numbers of products. For example, in the GAA case study with only three variants (Khajavirad et al. 2007), the all-in-one code failed to generate good results for the generalized case (see Fig. 11.9). This will be demonstrated numerically in Sect. 11.5.5.

Therefore, in order to show the benefit of generalized commonality, the all-in-one formulation was decomposed to ten sub-GAs, each optimizing an individual variant. The population size and maximum generation number are 3,000 and 1,400, respectively, which were verified as before.



**Fig. 11.10** Pareto fronts of the electric motor family for the generalized and all-or-none commonality definitions

**Table 11.3** Platform configuration of points on the Pareto frontier in Fig. 11.10

Modules	I	II	III
$N_c$	10	2, 4	2, 2
$N_s$	10	3, 7	4, 2, 2, 2
$A_{wa}$	10	9	5
$A_{wf}$	4, 4	4	4
$r_o$	5, 5	7	4
$T$	6	5, 2	4
$L$	10	2, 4	2, 2

The Pareto frontier for the generalized commonality is depicted in Fig. 11.10 and compared with the all-or-none case using the same decomposition scheme. As can be seen from the figure, relaxing the all-or-none restriction to generalized commonality among subsets of products improves the product family performance dramatically and allows an average 30 % of increased component sharing for the comparable level of performance leading to significant tooling cost savings.

In addition, to demonstrate the concept of generalized commonality, platform configurations for the three labeled points of the Pareto frontier are listed in Table 11.3. The numbers indicate the number of variants that share each component (1's are omitted). For instance, the notation {5, 2} indicates one design is shared among five variants while another design is common between two variants and the remaining three have distinct components. The presence of several sub-platforms for most of the components shows the importance of the generalization and also the efficiency of our new chromosome representation for capturing this feature.

### 11.5.4 Complexity of the Decomposition Scheme

To investigate the scalability of the proposed decomposition compared to the all-in-one problem, the electric motor example has been solved for different numbers of products. Pareto frontiers for 2, 4, 6, 8, and 10 variants are depicted in Fig. 11.11. Two graphs are shown for each case: The left hand shows the evolution of the decomposed algorithm, and the right hand compares the decomposed algorithm to the all-in-one approach. Specifically, the left hand graphs show the estimated Pareto front for several different values of MaxGen,<sup>11</sup> the maximum number of generations. Convergence was assumed when the average performance

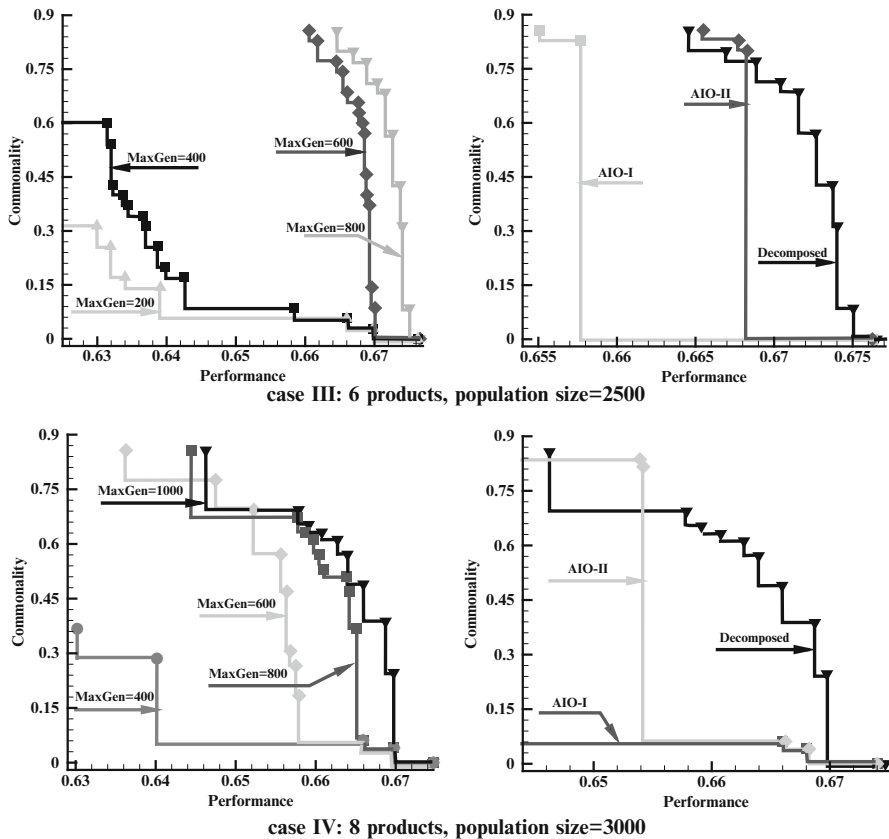


Fig. 11.11 Scalability of the proposed decomposition method versus the all-in-one problem

<sup>11</sup> Because of dynamic penalty function parameters for constraint handling that depend on the MaxGen parameter, the algorithm was restarted in each case from the same starting point.



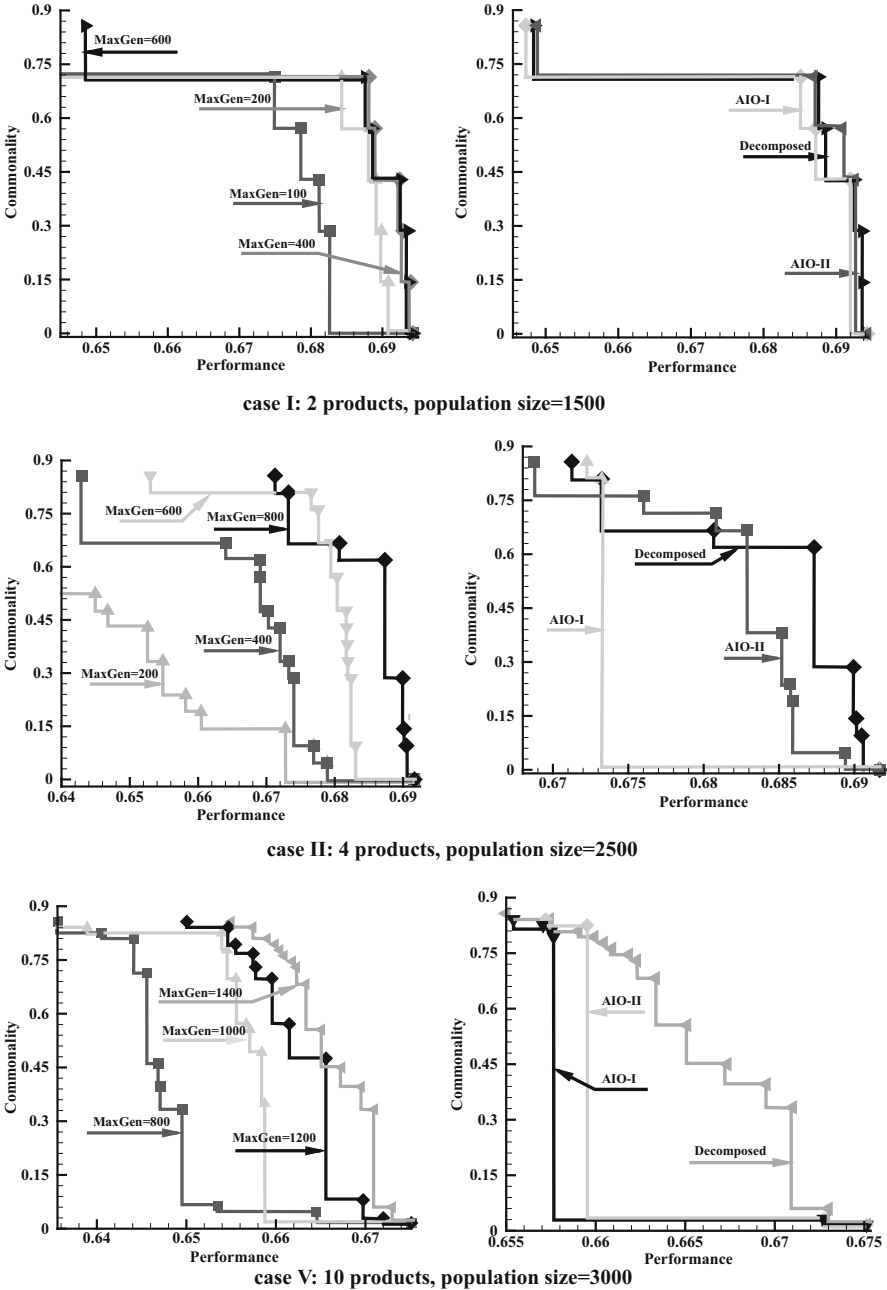
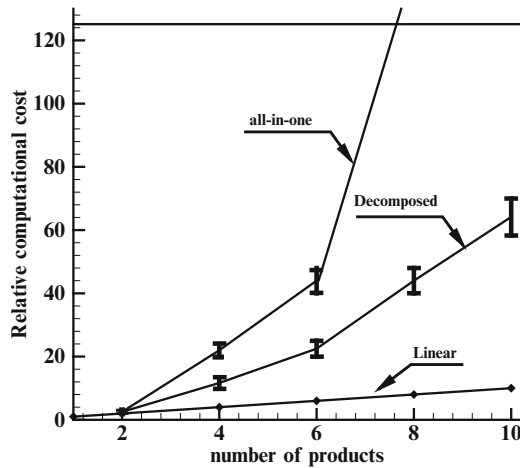


Fig. 11.11 (continued)



**Fig. 11.12** Complexity of the proposed method as a function of the number of variants (Relative computational cost for  $p$  products = No. of function evaluations for  $p$  products / No. of function evaluations for optimizing a single product)

improvement between two consecutive 200-generation steps was less than 1%. The right hand graphs compare the converged estimated Pareto front from the decomposed MOGA to two all-in-one cases: (1) AIO-I represents the results of running the all-in-one algorithm for an equal number of function evaluations, and (2) AIO-II represents the results of running the all-in-one algorithm until it is within 1% of the decomposed solution or has executed more than the maximum allowed function evaluations,<sup>12</sup> whichever is less. Meanwhile, Fig. 11.12 compares the computational cost required to achieve convergence for both the decomposed and the all-in-one algorithms. Error bars represent the 200-generation steps within which the 1% convergence criteria were achieved.

As can be observed from Figs. 11.11 and 11.12, as the number of variants in the family increases, the computational requirements increase exponentially for both methods. In the full decomposition scheme, increasing the number of variants has no effect on the size of each sub-GA; however, the 2D commonality chromosome size does grow, acting as the limiting factor for the algorithm scalability. This adverse effect could be seen in Fig. 11.12 by comparing the decomposed curve with the linear case. However, the relative benefit of decomposition over the all-in-one approach increases substantially as the number of variants increases, supporting improved scalability and the ability to solve larger problems than possible without decomposition.

<sup>12</sup>We set the maximum allowed number of function evaluations to twice the number of function evaluations required for solving the 10 product case using the decomposed algorithm.

**Table 11.4** Parallelization times for the motor example using various decomposition schemes

Decomposition scheme	Time per generation (s)		
	Computation time	Communication and idle time	Total execution time
2 sub-GAs (3 processors)	0.178	1.84	2.081
5 sub-GAs (6 processors)	0.081	3.38	3.461
10 sub-GAs (11 processors)	0.045	4.39	4.435

### 11.5.5 Parallelization

As shown in previous sections, decomposing the initial “all-in-one” formulation enhances the search quality of each sub-GA and decreases the total computational cost dramatically. Moreover, in order to further reduce the computational time, the decomposed approach can be parallelized by running each sub-GA on a separate processor and using the MPI library for exchanging data among different nodes. Total execution time of the parallel codes can be divided into three main parts: computation time, communication time (i.e., time for exchanging data among processors), and idle time (e.g., synchronization time). Hence, the final speedup depends on how these three factors change by increasing the number of processors, which is problem specific. In our decomposition scheme, the computational time for each sub-GA is inversely proportional to the number of sub-GAs. This is due to the fact that the number of genes in each sub-GA chromosome is inversely proportional to the total number of sub-GAs and the MOGA operators act on each gene separately. However, increasing the number of processors increases the communication and idle time. Hence, parallelization is beneficial for cases in which the computation time is the dominant part of the total execution time, which in the case of GAs is determined by the computational cost of the fitness calculation phase.

The computation, communication, and total execution time for one generation of the motor example are listed in Table 11.4 for 2, 5, and 10 sub-GAs (3, 6, and 11 processors), respectively. The reported times are the maximum time value among all the sub-GAs, i.e., the communication time and idle time are combined. As can be seen from the table, since each electric motor analysis (i.e., fitness calculation) involves only a number of analytic equations, and as result a very low computational cost ( $1.7 \times 10^{-6}$  s), total execution time is dominated by the communication time which is increased by increasing the number of processors. Hence, the total execution time is increased by increasing the number of processors. However, as can be found from Table 11.4, the computation time is reduced considerably by increasing the number of processors, and the communication time is negligible for cases involving time-consuming product-level simulations. Hence, parallelization is not recommended for the motor example; however, for more computationally intensive applications, a high speedup can be achieved through parallel processing.

## 11.6 Conclusions and Future Work

We introduced a new single-stage approach for solving the joint product family optimization problem using a unique decomposed MOGA formulation with a generalized commonality chromosome. The augmented chromosome representation introduced by Simpson and D'Souza (2004) was generalized to address component sharing among subsets of products. In order to improve the scalability of the proposed approach, the original all-in-one MOGA was decomposed into a novel two-level optimization problem in which the upper-level GA finds the optimal platform configuration while each lower-level GA optimizes a subset of products in the family. The proposed approach was demonstrated by optimizing a family of universal electric motors.

The effect of decomposition degree on the quality of estimated optimal fronts under fixed computational cost was investigated by solving the restricted commonality case using different decomposition schemes. The most decomposed case outperformed other schemes and found a well-converged and well-distributed optimal curve with a relatively low computational cost. Next, the same example was solved using the generalized commonality definition for the most decomposed case. Results show that the generalized commonality representation improves the optimal points dramatically, dominates all solutions of the all-or-none algorithm, and captures the tradeoff between commonality and performance more effectively. These trends are consistent with a second case study of a family of three general aviation aircraft, thus supporting generalizability of the empirical results. Finally, the complexity of the decomposition scheme in terms of number of variants was examined by solving families with different numbers of products; results show the proposed decomposition improves the scalability of the all-in-one problem significantly, extending the applicability of the optimization algorithm to families with more variants. More scalable algorithms could be explored by investigating alternatives to the proposed 2D commonality chromosome for representing the generalized component sharing.

In conclusion, common restrictions of commonality to degrees of all-or-none may significantly limit the quality of the resulting solutions and the ability to take full advantage of commonality options. Thus, we recommend that all-or-none restrictions be used only when the firm is truly uninterested in generalized commonality for reasons of logistics, etc. Secondly, the proposed decomposition is significantly more efficient than all-in-one approaches, and it is able to generate evenly distributed Pareto curves. We see no disadvantage to the decomposed approach, and we recommend the approach for future work in product line and product family optimization. In addition, for cases in which the fitness calculation phase (i.e., the product-level simulation) involves high computational cost, a high speedup can be achieved by running the decomposed approach on a parallel machine.

For future work, we intend to examine deterministic global optimization approaches to solve the joint product family problem (Khajavirad and Michalek 2008)

and compare the true optimal front with those found using popular heuristics and local solvers in product family optimization literature quantifying the benefits and limitations of each group. In addition, current objectives for commonality and deviation from exogenous performance targets used in this chapter, which are the standard in product family optimization, are somewhat artificial and limited proxies for the benefits of commonality and differentiation. Future work aims to introduce methods for quantifying cost benefits of commonality and revenue benefits of differentiation in a heterogeneous marketplace in order to make the most profitable tradeoff in product family planning and design (Kumar et al. 2006; Michalek et al. 2006).

**Acknowledgments** This work is supported in part by the Pennsylvania Infrastructure Technology Alliance, a partnership of Carnegie Mellon, Lehigh University, and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DCED). Dr. Simpson acknowledges support from the National Science Foundation under CAREER Award No. DMI-0133923. Any opinions, findings, and conclusions or recommendations presented in this chapter are those of the authors and do not necessarily reflect the views of the sponsors.

## References

- Akundi S, Simpson TW, Reed PM (2005) Multi-objective design optimization for product platform and product family design using genetic algorithms. In: ASME design engineering technical conferences, Long Beach, CA
- de Weck O (2005) Determining product platform extent. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design: methods and applications. Springer, New York, pp 241–301
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
- Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Parallel problem solving from nature VI conference, Paris, France, pp 849–858 [http://link.springer.com/chapter/10.1007%2F3-540-45356-3\\_83](http://link.springer.com/chapter/10.1007%2F3-540-45356-3_83)
- Fujita K (2002) Product variety optimization under modular architecture. *Comput Aid Des* 34 (12):953–965
- Fujita K, Yoshida H (2001) Product variety optimization: simultaneous optimization of module combination and module attributes. In: ASME design engineering technical conferences – design automation conference, Pittsburgh, PA
- Hassan R, de Weck O, Springmann P (2004) Architecting a communication satellite product line. In: 22nd AIAA international communications satellite systems conference and exhibit, Monterey, CA
- Khajavirad A, Michalek JJ (2007) A decomposed approach for solving the joint product family platform selection and design problem with generalized commonality. *ASME J Mech Des* 130:71–101
- Khajavirad A, Michalek JJ (2008) A deterministic Lagrangian-based global optimization approach for large scale decomposable problems. In: ASME international design engineering technical conferences and computers and information in engineering conference, ASME, New York
- Khajavirad A, Michalek JJ, Simpson TW (2007) A decomposed genetic algorithm for solving the joint product family optimization problem. In: 3rd AIAA multidisciplinary design optimization specialists conference, Honolulu, HI

- Khire RA, Messac A, Simpson TW (2006) Optimal design of product families using selection-integrated optimization (SIO) methodology. In: 11th AIAA/ISSMO symposium on multidisciplinary analysis and optimization, Portsmouth, VA
- Kokkolaras M, Fellini R, Kim HM, Michelena NF, Papalambros PY (2002) Extension of the target cascading formulation to the design of product families. *Struct Multidiscip Optim* 24:293–301
- Kumar D, Chen W, Simpson TW (2006) A market-driven approach to the design of platform-based product families. In: AIAA/ISSMO multidisciplinary analysis and optimization conference, Portsmouth, VA
- Martin M, Ishii K (1996) Design for variety: a methodology for understanding the costs of product proliferation. In: *Design theory and methodology*, Irvine, CA
- Messac A, Martinez MP, Simpson TW (2002) Effective product family design using physical programming. *Eng Optim* 34:245–261
- Michalek JJ, Ceryan O, Papalambros PY, Koren Y (2006) Balancing marketing and manufacturing objectives in product line design. *J Mech Des* 128(6):1196–1204
- Pacheco P (1997) *Parallel programming with MPI*. Morgan Kaufmann, San Francisco, CA
- Simpson TW (2005) Methods for optimizing product platforms and product families: overview and classification. In: Simpson TW, Siddique Z, Jiao J (eds) *Product platform and product family design: methods and applications*. Springer, New York, pp 133–156
- Simpson TW, D'Souza BS (2004) Assessing variable levels of platform commonality within a product family using a multiobjective genetic algorithm. *Concurr Eng Res Appl* 12:119–129
- Simpson TW, Maier JRA, Mistree F (1999) A product platform concept exploration method for product family design. In: *Design theory and methodology*, Las Vegas, NV
- Simpson TW, Maier JRA, Mistree F (2001) Product platform design: method and application. *Res Eng Des* 13:2–22
- Thevenot HJ, Simpson TW (2006) Commonality indices for product family design: a detailed comparison. *J Eng Des* 17:99–119

# Chapter 12

## One-Step Continuous Product Platform Planning: Methods and Applications

Achille Messac, Souma Chowdhury, and Ritesh Khire

**Abstract** This chapter presents two methodologies, Selection-Integrated Optimization (SIO) and Comprehensive Product Platform Planning (CP<sup>3</sup>), which convert the inherently combinatorial product family optimization problem into continuous optimization problems. These conversions enable one-step product family optimization without presuming the choice of platform and scaling design variables. Such approaches also enable taking full advantage of continuous optimization methods.

### 12.1 Introduction

In developing a successful product family, designers must translate the qualitative leveraging strategies into useful customer requirements to guide platform-based product development (Simpson et al. 2006). To this end, the engineering design community has developed and employed effective quantitative methods over the last two decades. The majority of these quantitative methods use some form of

---

Portions of this paper appeared in S. Chowdhury, A. Messac, and R. A. Khire (2011) Comprehensive Product Platform Planning (CP<sup>3</sup>) Framework, ASME Journal of Mechanical Design, 133(11), Paper No. 101004 (© ASME 2011), reprinted with permission.

A. Messac (✉)

Bagley College of Engineering, Mississippi State University, Mississippi State, MS 39762, USA

e-mail: [messac@bagley.msstate.edu](mailto:messac@bagley.msstate.edu)

S. Chowdhury

Department of Mechanical and Aerospace Engineering, Syracuse University, Syracuse, NY 13244, USA

R. Khire

United Technologies Research Center, East Hartford, CT 06118, USA

numerical optimization. A classification of these quantitative methods and a brief description of each class of methods are provided in the book-chapter by Simpson (2006). In this chapter, *product platform planning* refers to such quantitative design of a family of products.

Among the optimization-based *product family design* (PFD) methods, the class of methods that promises to avoid suboptimal solutions is the *single-stage approach with platform variables determined during optimization* (Khire et al. 2006). A brief summary of this class of methods is provided in the book-chapter by Simpson (2006). The quantification of the product design is generally a continuous process, whereas the platform identification is inherently a discrete process. The simultaneous (1) identification of platform and scaling design variables and (2) determination of the corresponding variable values are a challenging task (Khajavirad and Michalek 2008). The likely presence of complex nonlinear system functions further adds to this challenge. Powerful mixed-discrete nonlinear optimization methods need to be employed to address this challenge. As discussed in the book-chapter by Simpson (2006), genetic algorithms (GA) are suitable for this purpose.

An effective GA-based product family design method was developed by Khajavirad et al. (2009). In this GA-based method, a decomposition solution strategy is developed using the binary non-sorting genetic algorithm-II (NSGA-II) (Deb et al. 2002). This method provides flexibility in allowing the formation of a platform whenever a design variable (value) is shared by more than one product, and not necessarily all products in the family. This strategy eliminates the “*all or none*” restriction (Simpson 2006). The significant computational expense of the binary GA approach, especially in the case of large-scale problems, is addressed using a parallelized sub-GA solution strategy. Similar flexibility in platform creation is also presented by Chen and Wang (2008). The latter paper presents a PFD method that uses a 2-level chromosome genetic algorithm (2-LCGA) and proposes an information theoretical approach that incorporates fuzzy clustering and Shannon’s entropy to identify platform design variables. However, in this method, the process of platform creation precedes performance optimization of the product family. Consequently, the method developed by Chen and Wang (2008) exhibits the limiting attributes of the “two-step” approach. Another GA-based product family design method was presented by Jiao et al. (2007a), in which a generic method was developed to address various product family design scenarios. This method also included a hybrid constraint-handling technique to address complex and distinguishing constraints at different stages along the evolutionary process. The efficiency of the method developed by Jiao et al. (2007a) was illustrated by applying it to design a family of electric motors.

Product family design is a complex combinatorial optimization problem, which is also known to be intractable or NP-hard (Jiao et al. 2007b). The presence of discrete (integer) design variables appreciably increases the burden on the applied optimization method, particularly when one is dealing with large-scale real-life products (e.g., automobiles) and/or a large number of product variants (e.g., electric motors). In addition, the choice of optimization algorithms becomes limited when solving such a mixed-discrete optimization problem. Hence, an effective



continuous approximation of the single-step platform planning model could be expected to provide a welcome reprieve in this scenario. In the recent years, two such promising methods have been reported, which:

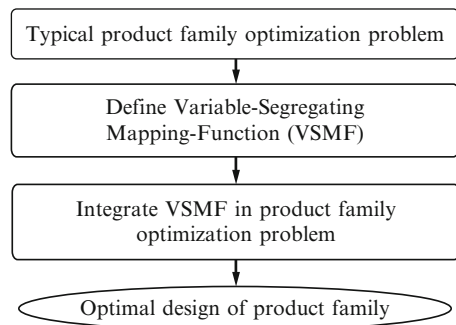
- Develop a tractable continuous approximation of the one-step product platform planning problem.
- Solve the approximated problem using standard continuous optimization methods.

This chapter presents the development of the two *one-step continuous PFD approaches*: (1) the Selection-Integrated Optimization [SIO (Khire 2006)] and (2) the Comprehensive Product Platform Planning [CP<sup>3</sup> (Chowdhury et al. 2011)]. Section 12.2 provides the formulations of the integrated product platform planning models in these two approaches. Section 12.3 presents the continuous optimization strategies developed and implemented in these two approaches. Section 12.4 discusses the results from the application of these approaches to design families of universal electric motors. Closing remarks are offered in Sect. 12.5.

## 12.2 Integrated Product Planning Models

### 12.2.1 Selection-Integrated Optimization (SIO) Model

The SIO method, introduced by Khire and Messac (2008), solves a continuous approximation of the product family design problem. A penalty function is formulated to represent the lack of commonality among products. Subsequently, product family optimization is performed by implementing the *Variable-Segregating Mapping-Function* (VSMF) scheme in the order shown in Fig. 12.1. The motivation of the SIO methodology is to eliminate a significant source of suboptimality involved in the adaptive system optimization (or product family design) procedure caused by the separation of the (1) selection and (2) optimization processes. The next section discusses the development of the VSMF strategy and



**Fig. 12.1** Main steps in the application of SIO methodology in PFD

how it is implemented within the SIO framework to create and optimize a continuous approximation of the product family design problem.

The example of the universal electric motor family is used to illustrate the formulation and the application of the SIO method. As seen from the book-chapter by Simpson (2006), the design configuration of the motor changes to satisfy different output torque requirements. For the motor, a change in the design configuration involves the scaling of the following design variables: cross-sectional area of the armature ( $A_{wa}$ ), cross-sectional area of the field pole wire ( $A_{wf}$ ), number of wire turns on each field pole ( $N_s$ ), number of wire turns on the armature ( $N_c$ ), stack length of the motor ( $L$ ), radius of the stator ( $r_0$ ), and thickness of the stator ( $t$ ). The scaling of these design variables is likely to incur an additional manufacturing cost because of factors such as complex tooling requirements. These additional factors contribute to the penalty, which is measured in terms of *the difference in the values of the above scaling variables*.

### 12.2.1.1 Penalty Function Formulation

The penalty function measure is applicable mostly in the case of scale-based product families; it is defined in terms of the design variables. Reducing the penalty function seeks to increase the commonality among the products in a family. Mathematically, the penalty function,  $f_{\text{pen}}$ , is expressed as the summation of ratios of the standard deviation and mean values of each design variable within the product family, as given by

$$f_{\text{pen}} = \frac{\sigma_{x_1}}{\mu_{x_1}} + \frac{\sigma_{x_2}}{\mu_{x_2}} + \dots + \frac{\sigma_{x_n}}{\mu_{x_n}} \quad (12.1)$$

where  $n$  is the number of design variables that participate in platform planning, the parameter  $\sigma_{x_i}$  represents the standard deviation of the generic  $i$ th design variable across all the product variants, and  $\mu_{x_i}$  represents the mean of the generic  $i$ th design variable across all the product variants. In the case of the motor family, the penalty function is therefore expressed as

$$f_{\text{pen}} = \frac{\sigma_{A_{wa}}}{\mu_{A_{wa}}} + \frac{\sigma_{A_{wf}}}{\mu_{A_{wf}}} + \frac{\sigma_{N_s}}{\mu_{N_s}} + \frac{\sigma_{N_c}}{\mu_{N_c}} + \frac{\sigma_L}{\mu_L} + \frac{\sigma_{r_0}}{\mu_{r_0}} + \frac{\sigma_t}{\mu_t} \quad (12.2)$$

where the subscripts indicate which variable the standard deviation and the mean parameters are associated with—e.g.,  $\sigma_L$  represents the standard deviation of the motor stack length across all the motor variants and  $\mu_L$  represents the mean of the motor stack length across all the motor variants. It is important to note that this penalty function can be further advanced by incorporating weights for the terms corresponding to each design variable (a weighted sum). The minimization of the above penalty function results in the minimization of the standard deviations, which encourages the design variables to become a part of the platform.

### 12.2.1.2 SIO Problem Formulation

In the bi-objective problem for the electric motor product family (with  $N$  motor variants), the performance and the penalty objective functions are combined into an aggregate objective function (AOF). The optimization problem is formulated as

$$\begin{aligned}
 & \text{Min} \quad w_1 f_{\text{per}}(X) + w_2 f_{\text{pen}}(X) \\
 & \text{subject to} \\
 & \quad T^k(X) = T_{rq}^k \\
 & \quad P_{\text{out}}^k(X) = 300 \text{ N/m} \\
 & \quad M_{\text{total}}^k(X) \leq 2 \text{ kg} \\
 & \quad H^k(X) \leq 5000 \text{ Amp.turns/m} \\
 & \quad \eta^k(X) \geq 0.15 \\
 & \quad \frac{r_o^k}{t^k} \geq 1 \\
 & \quad X^k = [N_c \quad N_s \quad A_{wa} \quad A_{wf} \quad r_o \quad t \quad L \quad I]^T \\
 & \quad \forall k = 1, 2, \dots, N
 \end{aligned} \tag{12.3}$$

where  $T^k$ ,  $P_{\text{out}}^k$ ,  $M_{\text{total}}^k$ ,  $H^k$ , and  $\eta^k$ , respectively, represent the torque, the power output, the total mass, the magnetization intensity, and the efficiency of the  $k$ th motor;  $w_1$  and  $w_2$  are the weights for the performance and the penalty objectives, respectively; and  $f_{\text{per}}$  is the overall performance objective of the family. In the case of the motor family, Khire (2006) used the average of the individual motor efficiencies as the performance objective:

$$f_{\text{per}} = - \sum_{i=1}^N \frac{\eta_i}{N} \tag{12.4}$$

## 12.2.2 Comprehensive Product Platform Planning (CP<sup>3</sup>) Model

Several existing PFD methodologies [including recent methods (Chen and Wang 2008)] do not readily represent the platform planning process by a mathematical model that is *independent* of the optimization process. In other words, the models are formulated such that a particular class of optimization algorithms must be leveraged to effectively optimize the PFD. The development of generalized models of the platform planning process would provide the leverage to choose from different classes of optimization algorithms to solve the problem; this choice practically depends on the complexity of the concerned system and the user's experience with

particular optimization methods. Moreover, a generalized mathematical model can facilitate helpful investigation of the underlying mathematics (complex and combinatorial) of product platform planning.

Existing PFD models also tend to make a clear distinction between scale-based and module-based product families. From Table 8 in the book-chapter by Simpson (2006), it can be seen that a majority of the existing PFD methods are reported to be suitable for “*either scale-based or module-based*” product families. The scale-based PFD methods assume that each product is comprised of all the design variables involved in the family; as a result, these methods do not readily apply to modular product families. On the other hand, typical modular PFD methods often cannot readily account (without platform/scaling assumptions) for the simultaneous presence of modular and scalable product attributes. The simultaneous presence of modular and scalable product attributes is common for complex real-life systems/products—e.g., in a laptop, the DVD drive can have both modular (DVD, DVD-RW, Blue-ray Disc) and scalable (drive speeds =  $1\times$ ,  $2\times$ , ...,  $N\times$  1.35 MB/s) properties.

The Comprehensive Product Platform Planning (CP<sup>3</sup>) method, developed by Chowdhury et al. (2011), formulates a generalized model that is (1) independent of the eventual solution strategy and (2) seeks to avoid traditional distinctions between modular and scalable families. This model not only allows effective design optimization of product families but also promotes further investigation of the underlying processes in product platform planning, e.g., (1) how sensitive are the commonality indices to the number of product variants in each platform and (2) how the production volume of each product variant affects optimal product platform planning. Other important features of the CP<sup>3</sup> model are:

- This model avoids the “all or none” restriction (Simpson 2006); this approach thus allows the formation of subfamilies.
- This model facilitates simultaneous (1) selection of platform/scaling design variables and (2) optimization of the physical design of the product variants.
- This model yields a mixed-integer nonlinear programming problem (MINLP) for the optimization process.

In the subsequent sections, the formulation of the CP<sup>3</sup> model is presented and is followed by an illustration of this model using the example of a representative “4 products-5 variables” family. This section is concluded with the definition of the overall optimization problem yielded by the CP<sup>3</sup> model.

### 12.2.2.1 Formulation of the CP<sup>3</sup> Model

The CP<sup>3</sup> model is founded on the concept of a mixed-integer nonlinear equality constraint (Chowdhury et al. 2011). This formulation is illustrated using a representative family of two products that are comprised of three design variables each. Table 12.1 shows the variables involved in this example. For a design variable,  $x_{kj}$ , in Table 12.1, the superscript ( $k$ ) and the subscript ( $j$ ) represent the

**Table 12.1** A representative family of two products (© ASME 2011), reprinted with permission

Physical design variable	Product 1	Product 2	Integer variables
1st variable	$x_1^1$	$x_1^2$	$\lambda_1^{12}$
2nd variable	$x_2^1$	$x_2^2$	$\lambda_2^{12}$
3rd variable	$x_3^1$	$x_3^2$	$\lambda_3^{12}$

product number and the variable number, respectively. The binary-integer variables (represented by  $\lambda$ ) given in Table 12.1 are defined as

$$\lambda_j^{12} = \begin{cases} 1, & \text{if } x_j^1 = x_j^2 \\ 0, & \text{otherwise} \end{cases} \tag{12.5}$$

Hence, the  $\lambda$  variables determine the commonality among products, with respect to the system design variables. The general MINLP problem, formulated to represent the design optimization of the product family example (shown in Table 12.1), is given by

$$\begin{aligned} & \text{Max } f_{\text{perf}}(Y) \\ & \text{Min } f_{\text{cost}}(Y) \\ & \text{s.t. } \lambda_1^{12}(x_1^1 - x_1^2)^2 + \lambda_2^{12}(x_2^1 - x_2^2)^2 + \lambda_3^{12}(x_3^1 - x_3^2)^2 = 0 \\ & \quad g_i(X) \leq 0, \quad i = 1, 2, \dots, p \\ & \quad h_i(X) = 0, \quad i = 1, 2, \dots, q \\ & \quad Y = \{x_1^1, x_2^1, x_3^1, x_1^2, x_2^2, x_3^2, \lambda_1^{12}, \lambda_2^{12}, \lambda_3^{12}\} \\ & \quad X = \{x_1^1, x_2^1, x_3^1, x_1^2, x_2^2, x_3^2\} \\ & \quad (\lambda_1^{12}, \lambda_2^{12}, \lambda_3^{12}) \in B : B = \{0, 1\} \end{aligned} \tag{12.6}$$

and where  $f_{\text{perf}}$  and  $f_{\text{cost}}$  are the objective functions that represent the performance and the cost of the product family, respectively. In Eq. (12.6),  $g_i$  and  $h_i$ , respectively, represent the inequality and equality constraints related to the physical design of the product. The first equality constraint in Eq. (12.6), which involves the generic parameters  $\lambda_j^{kl}$ , is termed the commonality constraint. This constraint can be represented in a compact matrix format as

$$\begin{bmatrix} x_1^1 & x_1^2 & x_2^1 & x_2^2 & x_3^1 & x_3^2 \end{bmatrix} \begin{bmatrix} \lambda_1^{11} & -\lambda_1^{12} & 0 & 0 & 0 & 0 \\ -\lambda_1^{21} & \lambda_1^{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_2^{11} & -\lambda_2^{12} & 0 & 0 \\ 0 & 0 & -\lambda_2^{21} & \lambda_2^{22} & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_3^{11} & -\lambda_3^{12} \\ 0 & 0 & 0 & 0 & -\lambda_3^{21} & \lambda_3^{22} \end{bmatrix} \begin{bmatrix} x_1^1 \\ x_1^2 \\ x_2^1 \\ x_2^2 \\ x_3^1 \\ x_3^2 \end{bmatrix} = 0 \tag{12.7}$$

This formulation can be extended to a general product family, comprising  $N$  products and  $n$  design variables, as given by

$$X^T \Lambda X = 0$$

$$\Lambda = \begin{bmatrix} \sum_{k \neq 1} \lambda_1^{1k} & \cdots & -\lambda_1^{1N} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\lambda_1^{N1} & \cdots & \sum_{k \neq N} \lambda_1^{Nk} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \sum_{k \neq 1} \lambda_j^{1k} & \cdots & -\lambda_j^{1N} & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & -\lambda_j^{N1} & \cdots & \sum_{k \neq N} \lambda_j^{Nk} & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sum_{k \neq 1} \lambda_n^{1k} & \cdots & -\lambda_n^{1N} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda_n^{N1} & \cdots & \sum_{k \neq N} \lambda_n^{Nk} \end{bmatrix}$$

$$k = 1, 2, \dots, N$$

$$X = [x_1^1 \ x_1^2 \ \cdots \ x_1^N \ \cdots \ x_j^1 \ x_j^2 \ \cdots \ x_j^N \ \cdots \ x_n^1 \ x_n^2 \ \cdots \ x_n^N]^T \tag{12.8}$$

The matrix  $\Lambda$  is called the commonality constraint matrix. This matrix is a symmetric block diagonal matrix, where the  $j$ th block corresponds to the  $j$ th design variable. An explicit representation of each block of the  $\Lambda$  matrix is given by

$$\Lambda_j = \begin{bmatrix} \sum_{k \neq 1} \lambda_j^{1k} & -\lambda_j^{12} & \cdots & -\lambda_j^{1l} & \cdots & -\lambda_j^{1N} \\ -\lambda_j^{21} & \sum_{k \neq 2} \lambda_j^{2k} & \cdots & -\lambda_j^{2l} & \cdots & -\lambda_j^{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -\lambda_j^{l1} & -\lambda_j^{l2} & \cdots & \sum_{k \neq l} \lambda_j^{lk} & \cdots & -\lambda_j^{lN} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -\lambda_j^{N1} & -\lambda_j^{N2} & \cdots & -\lambda_j^{Nl} & \cdots & \sum_{k \neq N} \lambda_j^{Nk} \end{bmatrix} \tag{12.9}$$

The commonality constraint matrix can be derived from the generalized commonality matrix  $\lambda$  that is expressed as

$$\lambda = \begin{bmatrix} \lambda_1^{11} & \dots & \lambda_1^{1N} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_1^{N1} & \dots & \lambda_1^{NN} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \lambda_j^{11} & \dots & \lambda_j^{1N} & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \lambda_j^{N1} & \dots & \lambda_j^{NN} & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_n^{11} & \dots & \lambda_n^{1N} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_n^{N1} & \dots & \lambda_n^{NN} \end{bmatrix} \quad (12.10)$$

$$\underbrace{\lambda_j^{kl}}_{k \neq l} = \begin{cases} 1, & \text{if } \lambda_j^{kk} = \lambda_j^{ll} = 1 \text{ and } x_j^k = x_j^l \\ 0, & \text{otherwise} \end{cases}$$

$$\lambda_j^{kk} = \begin{cases} 1, & \text{if } j^{\text{th}} \text{ variable is included in product-}k \\ 0, & \text{if } j^{\text{th}} \text{ variable is NOT included in product-}k \end{cases}$$

It can be observed from Eq. (12.10) that the commonality matrix is also a symmetric block diagonal matrix. The off-diagonal elements of the commonality matrix ( $\lambda_j^{kl}$ ) are called the *commonality variables* in the remainder of this chapter. The diagonal elements of the commonality matrix ( $\lambda_j^{kk}$ ) determine whether the  $j$ th variable is included in product- $k$ . This commonality matrix definition is similar to that presented by Khajavirad and Michalek (2008). Every block of the constraint commonality matrix can be expressed as a function of the corresponding commonality matrix block:

$$A_j = f_{con}(\lambda_j) \quad (12.11)$$

In typically modular product families, different products might comprise different types and different numbers of modules. Each module is comprised of a particular set of design variables that are also known as module attributes; these attributes may be shared by more than one module in a complex system. Consequently, different products can be comprised of physically different sets of design variables. In order

to address a modular product family design, a product platform planning model should account for the inclusion, the exclusion, and the substitution of design variables. These three possibilities can be captured by the novel commonality matrix. In this context, one of the following three distinct scenarios can occur:

1. The  $j$ th design variable is required in product- $k$ ; consequently,  $\lambda_j^{kk} = 1$  would be known a priori (fixed).
2. The  $j$ th design variable is not required for product- $k$ ; consequently,  $\lambda_j^{kk} = 0$  would be known a priori (fixed).
3. Inclusion of the  $j$ th design variable is optional for product- $k$ ; the corresponding  $\lambda_j^{kk}$  element is allowed to be determined during the course of product family optimization (treated as a variable).

If the third scenario occurs, the CP<sup>3</sup> model does not make any prior assumptions regarding the attribute values or whether a module (or the corresponding attributes) is shared by multiple products (forms a platform or not). If the first scenario occurs, the module attributes themselves may scale among the products that must include them. The commonality matrix blocks representing the scaling design variables (attributes included in all product variants) should have the diagonal elements fixed at one. Therefore, careful specification/management of the diagonal elements of the commonality matrix automatically addresses the modularity/scalability properties of the design variables, without imposing limiting distinctions. This model allows a coherent consideration of the likely combination of scaling and modular attributes in a product family, which is uniquely helpful.

To further explain the helpful capability of the CP<sup>3</sup> model to flexibly address combined modular-scaling families, a family of laptops example is considered. Three laptops (a 15 in., a 13 in., and a netbook) are being designed, and the manufacturer decides that the 15 in. will have a DVD drive, the 13 in. might/might not have a DVD drive (based on the ensuing overall product cost), and the netbook will not have a DVD drive (due to space constraints). A key constituent design variable of the DVD-drive module is the drive speed/data rate that is a scaling attribute: e.g., “ $1 \times$  or  $2 \times 1.35$  MB/s.” In this case, the “ $3 \times 3$ ” commonality matrix block corresponding to this design variable (drive speed/data rate) can be expressed as

$$\lambda_{\text{DVD}} = \begin{bmatrix} 1 & \lambda^{12} & 0 \\ \lambda^{21} & \lambda^{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (12.12)$$

where the order of the products is [15 in., 13 in., netbook] and the commonality variables  $\lambda^{12}$ ,  $\lambda^{21}$ , and  $\lambda^{22}$  are to be determined during optimization. This commonality matrix formulation (Eq. 12.10) thus provides a generalized product platform planning model, which can address a wide variety of product families without making limiting distinctions between scalable and modular attributes.



**Table 12.2** Sample product platform plan for a family of four products and five variables

Physical design variables	Product 1	Product 2	Product 3	Product 4
1st variable ( $x_1$ )	A	A	A	A
2nd variable ( $x_2$ )	—	—	—	—
3rd variable ( $x_3$ )	B	B	C	C
4th variable ( $x_4$ )	—	—	—	—
5th variable ( $x_5$ )	D	—	—	D

It is helpful to note that factors, such as (1) the product-module architecture and (2) the availability of module options in a modular family, demand additional considerations which are not explicitly addressed by the current CP<sup>3</sup> model. For example, in the case of laptops, technologically differing/substitutable types of DVD drives are commercially available: e.g., DVD, DVD-RW, and Blue-ray Disc. Although, these DVD-drive options have the same drive-speed attribute, the commonality representation in the CP<sup>3</sup> model would not readily apply if two laptop variants were to use two different DVD-drive types. Moreover, the modules in a product are often not independent of each other from a design and/or operation perspective; in that case, inclusions/exclusion/substitution of modules can be mutually dependent (e.g., web camera and microphone), which leads to dependent commonality matrix blocks. Such “module interdependency” that can be defined within the conceived product architecture is not addressed by the current CP<sup>3</sup> model. Appropriate considerations of the underlying product architecture and module options should further advance the applicability of the CP<sup>3</sup> model and are considered a key area for future research.

### 12.2.2.2 Representative Illustration of the CP<sup>3</sup> Model

The proposed CP<sup>3</sup> model is illustrated using an example of a product family comprising four products. The entire family has an exhaustive set of five different variables. It is helpful at this point to provide a careful definition of a generalized product platform—“*A product platform is said to be created when more than one product variant in a family share a particular design variable.*” In this case “*sharing a design variable between two products*” can be defined as the products having the same value of the concerned design variable. Table 12.2 shows the platform plan of the sample product family. Each uppercase letter in Table 12.2, except the “—” symbol, represents a platform. Blocks in Table 12.2, displaying similar letters, imply that the corresponding products are members of a particular variable-based platform. A block displaying the “—” symbol represents non-platform (scaling) design variable values, which implies that the corresponding design variable value is not shared by more than one product. Blocks without any specified symbol/letter (blank) imply that the particular variable is not included in the corresponding product.

A product platform plan, as shown in the example (in Table 12.2), entails classifying design variables (in the entire family) into the following three categories:

1. *Platform design variable*: A design variable that is shared by all the different kinds of products in the family, e.g., variable  $x_1$  in Table 12.2.
2. *Sub-platform design variable*: A design variable that is shared by a particular subset of the different kinds of products in the family, leading to subfamily, e.g., variable  $x_5$  in Table 12.2. Sub-platforms may also lead to multiple subfamilies with respect to a design variable, e.g., variable  $x_3$  in Table 12.2.
3. *Non-platform design variable*: A design variable that is not shared by more than one product in the family, e.g., variables  $x_2$  and  $x_4$  in Table 12.2.

The diagonal blocks of the *commonality matrix*, corresponding to each design variable for the product family illustrated in Table 12.2, are given by

$$\begin{aligned} \lambda_1 &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \lambda_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \lambda_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \\ \lambda_4 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \lambda_5 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (12.13)$$

The corresponding five diagonal blocks of the *commonality constraint matrix* ( $\Lambda$ ), determined from the *commonality matrix*, are given by

$$\begin{aligned} \Lambda_1 &= \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}, \Lambda_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \Lambda_1 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \\ \Lambda_1 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \Lambda_1 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (12.14)$$

### 12.2.2.3 The Generalized MINLP Problem

The generalized MINLP problem for a family of  $N$  products, comprising a global set of  $n$  design variables, can be expressed as

$$\begin{aligned}
 & \text{Max } f_{\text{perf}}(Y) \\
 & \text{Min } f_{\text{cost}}(Y) \\
 & \text{subject to} \\
 & X^T \Lambda X = 0 \\
 & g_i(X) \leq 0, \quad i = 1, 2, \dots, p \\
 & h_i(X) = 0, \quad i = 1, 2, \dots, q \\
 & \Lambda = f_{\text{con}}(\lambda) \\
 & Y = \{X, \lambda\} \\
 & X = [x_1^1 \quad x_1^2 \quad \cdots \quad x_1^N \quad \cdots \quad x_j^1 \quad x_j^2 \quad \cdots \quad x_j^N \quad \cdots \quad x_n^1 \quad x_n^2 \quad \cdots \quad x_n^N]^T \\
 & \lambda_j^{kl} \in B : B = \{0, 1\} \\
 & k, l = 1, 2, \dots, N; \quad j = 1, 2, \dots, n
 \end{aligned} \tag{12.15}$$

and where the matrices  $\Lambda$  and  $\lambda$  are given by Eqs. 12.8 and 12.10, respectively. It is helpful to note that, although the matrix  $\lambda$  is a variable for the MINLP problem, certain diagonal elements ( $\lambda_l^{kk}$ ) are known a priori (see Sect. 12.2).

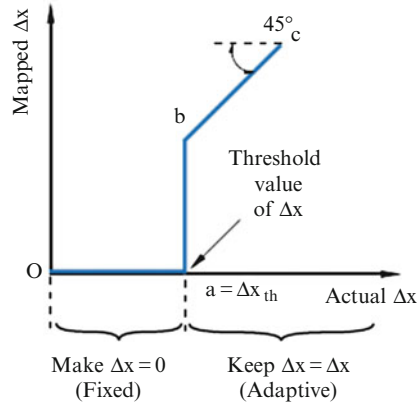
## 12.3 One-Step Continuous Optimization Frameworks

### 12.3.1 Product Family Design Using Selection-Integrated Optimization (SIO)

The SIO method was introduced by Khire and Messac (2008) primarily to solve the optimization of adaptive systems. The Variable-Segregating Mapping Function (VSMF) provides a continuous approximation of the discrete problem posed by adaptive systems. In adaptive system optimization, the objective is to minimize the change/variation in the design variables while maximizing the overall system performance. The analogy between adaptive system design and product family design is evident (Khire 2006). A product family design problem can therefore be reformulated into a similar problem, where the penalty function to be minimized represents the overall variance in the design variables in the product family.

In conventional two-step PFD approaches, if the difference in the values of a design variable (among the product variants),  $\Delta x_k$ , is smaller than a prespecified

**Fig. 12.2** Selecting design variables to be fixed (platform) or made adaptive (scalable)



threshold difference (Fellini et al. 2005),  $\Delta x_k^{th}$ , the corresponding design variable ( $\Delta x_k$ ) is fixed across all product variants (a platform variable); otherwise, it is made scalable. Figure 12.2 provides a graphical representation of this mapping.

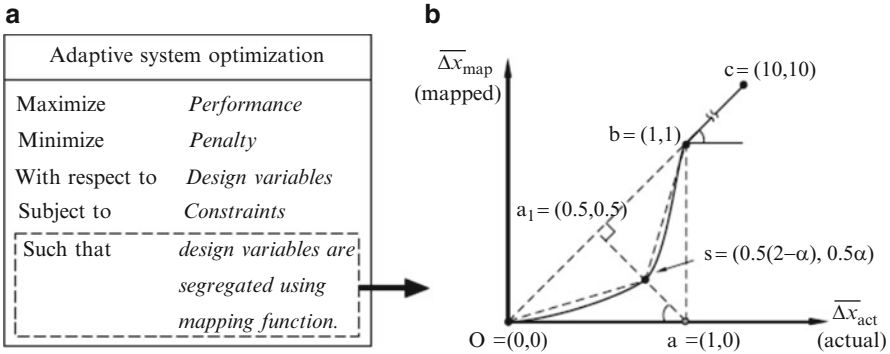
The conventional mapping is not continuous and therefore is not suitable for tractable application of gradient-based algorithms or other standard continuous optimization strategies. More importantly, the threshold value ( $\Delta x_k^{th}$ ) is usually heuristically chosen, which again introduces a source of suboptimality. The Variable-Segregating Mapping Function (VSMF) overcomes these two issues. The VSMF facilitates the automatic segregation of the platform variables from the scaling design variables. By including the VSMF in the optimization problem, the SIO methodology segregates the design variables in the course of the optimization process.

**12.3.1.1 Variable-Segregating Mapping Functions (VSMF)**

The Variable-Segregating Mapping Function (VSMF) is a family of continuous functions that progressively approximates the discontinuous mapping shown in Fig. 12.2. A generic VSMF is defined for all design variables. Details of the VSMF are provided in Fig. 12.3b. The VSMF is defined in terms of two normalized and nondimensional variables: (1)  $\overline{\Delta x}_{act}$ , representing *actual variation*, and (2)  $\overline{\Delta x}_{map}$ , representing *mapped variation*. In the case of PFD, *variation* represents the difference between the highest and the lowest values of a design variable among the product variants.

The VSMF is defined such that it satisfies the following properties:

1. It is a monotonically increasing smooth function (continuous first derivative).
2. The threshold value is set at  $\overline{\Delta x}_{act} = 1$ , at point a in Fig. 12.3b.
3. At  $\overline{\Delta x}_{act} = 0$  (point O),  $\overline{\Delta x}_{map} = 0$ .
4. At  $\overline{\Delta x}_{act} = 1$  (point b),  $\overline{\Delta x}_{map} = 1$ .



**Fig. 12.3** Application of the SIO method for PFD. (a) Overall formulation. (b) Graphical representation of VSMF

5. For  $\overline{\Delta x}_{act} \geq 0$  (segment b-c),  $\overline{\Delta x}_{map} = \overline{\Delta x}_{act}$ .
6. The VSMF contains a point  $s$  between  $\overline{\Delta x}_{act} = 0$  and  $\overline{\Delta x}_{act} = 1$ ; this point has an interesting and important property, which is discussed next.

The point  $s$  divides the VSMF into two parts, depicted by O- $s$  and  $s$ -b in Fig. 12.3b. The coordinates of point  $s$  are governed by a special parameter  $\alpha$  as given by

$$s = [0.5(2 - \alpha), 0.5\alpha] \tag{12.16}$$

By varying  $\alpha$  between 1 and 0, a family of VSMFs can be obtained—a property exploited for segregating the fixed from the adaptive design variables. It is important to note that  $\alpha$  is not a design variable in the SIO methodology and is instead a VSMF parameter that facilitates the progressive approximation of the discreteness involved in the design variable selection process. For  $\alpha = 1$ , the VSMF follows the straight line O- $a_1$ -b-c in Fig. 12.3b. If the value of  $\alpha$  is progressively lowered towards zero, point  $s$  travels from point  $a_1$  to point  $a$ , thereby causing the VSMF to progressively approximate the original discontinuous mapping depicted by O-a-b-c in Fig. 12.3b. This progression bears a significant similarity to homotopy-based approaches (Watson and Haftka 1989).

The point  $s$  can be called the *separating point*. Based on this point, Khire (2006) proposed the following segregation criterion:

$$\text{if } \lim_{\alpha \rightarrow 0} (\overline{\Delta x}_k)_{map} \leq 0.5, \text{ then } x_k \text{ is fixed} \tag{12.17}$$

where  $0.5\alpha$  is the coordinate of the separating point  $s$  along the vertical axis, which vanishes as  $\alpha$  goes to zero. If the condition in Eq. (12.17) is not satisfied, then the design variable ( $x_k$ ) is considered to be a scaling variable.

### 12.3.1.2 Implementation of VSMF in SIO-Based PFD

In order to implement VSMF, the penalty function is redefined as

$$\begin{aligned} \bar{f}_{\text{pen}} = & \left[ (\overline{\Delta A_{wa}})_{\text{map}} - 0.5\alpha \right] + \left[ (\overline{\Delta A_{wf}})_{\text{map}} - 0.5\alpha \right] \\ & + \left[ (\overline{\Delta N_s})_{\text{map}} - 0.5\alpha \right] + \left[ (\overline{\Delta N_s})_{\text{map}} - 0.5\alpha \right] \\ & + \left[ (\overline{\Delta L})_{\text{map}} - 0.5\alpha \right] + \left[ (\overline{\Delta r_0})_{\text{map}} - 0.5\alpha \right] + \left[ (\overline{\Delta t})_{\text{map}} - 0.5\alpha \right] \end{aligned} \quad (12.18)$$

where each variable with the bar overhead represents the mapped variation in the corresponding motor design variable. The above defined penalty objective function is used as one of the objectives in the bi-objective PFD problem presented in Eq. (12.3), with the design variables modified according to the VSMF:

$$(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_7) = \text{VSMF}(\alpha, x_1, x_2, \dots, x_7) \quad (12.19)$$

For the  $k$ th variable, the VSMF-based modification defined in Eq. (12.19) is performed using the following set of expressions:

$$\Delta x_k = x_k^{\max} - x_k^{\min}$$

where

$$x_k^{\max} = \max(x_k^1, x_k^2, \dots, x_k^{10}); \quad x_k^{\min} = \min(x_k^1, x_k^2, \dots, x_k^{10}) \quad (12.20)$$

$$(\overline{\Delta x_k})_{\text{act}} = \frac{\Delta x_k}{(\Delta x_k)_{\text{th}}} \quad (12.21)$$

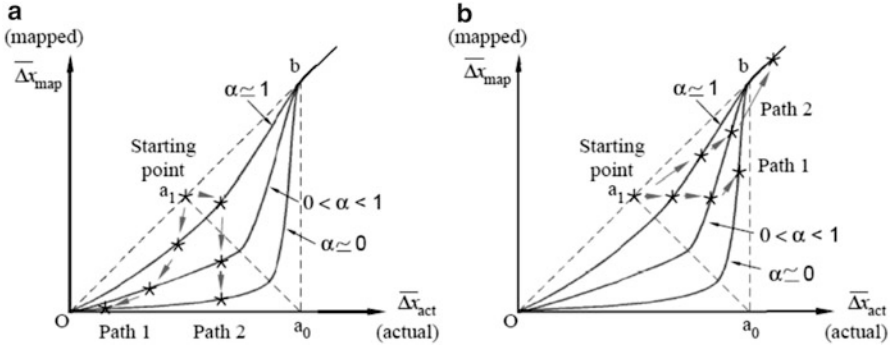
$$\tilde{x}_k^j = x_k^{\min} + \frac{(\overline{\Delta x_k})_{\text{map}}}{(\overline{\Delta x_k})_{\text{act}}} (x_k^j - x_k^{\min}) \quad (12.22)$$

In Eq. (12.21),  $(\Delta x_k)_{\text{th}}$  is the threshold value of the  $k$ th variable, which can be defined as

$$(\Delta x_k)_{\text{th}} = 2\Delta x_k^* = 2(x_k^{\max^*} - x_k^{\min^*}) \quad (12.23)$$

where  $x_k^{\max^*}$  and  $x_k^{\min^*}$ , respectively, represent the highest and the lowest values of the  $k$ th variable among the ten motor variants, after each product has been individually optimized for maximum performance. In Eq. (12.22),  $(\overline{\Delta x_k})_{\text{map}}$  is estimated from  $(\overline{\Delta x_k})_{\text{act}}$  using the generic VSMF mapping, as defined in the previous section.

In SIO-based PFD, the optimal product platform plan is obtained by an iterative solution process, where the bi-objective optimization problem (given by Eq. 12.3) is solved at each iteration. The value of  $\alpha$  is lowered by 0.1 at the beginning of each iteration except the first one (i.e.,  $\Delta\alpha = 0.1$ ), starting with  $\alpha = 0.9$  and ending at  $\alpha = 0.1$  (i.e., ten iterations). The optimization solution from the previous iteration is



**Fig. 12.4** Segregation of design variable using generic VSMF (© ASME 2011), reprinted with permission. (a) Segregation of platform variables. (b) Segregation of scaling variables

used as the starting guess for the design variables in the current iteration, which are subsequently modified using the VSMF. The process of identifying platform and scaling variables using the VSMF-based SIO method is illustrated in Fig. 12.4.

In Fig. 12.4, the stars show the typical locations of the optimal solutions obtained from each repetition of the optimization problem. As shown in Fig. 12.4a, as the value of  $\alpha$  is lowered in each repetition, the design variables that are going to be fixed (platform) move closer to segment O- $a_0$  on their corresponding VSMFs. We note that on segment O- $a_0$ , the mapped change in the design variable is zero and therefore represents a fixed design variable. Also, with each repetition, the scaling design variables move closer to or further than point  $b$  (see Fig. 12.4b). Hence, the SIO methodology segregates the platform variables from the scaling variables within the optimization process, thereby allowing the optimality of the resulting product family design.

### 12.3.2 Comprehensive Product Platform Planning (CP<sup>3</sup>) Optimization

The combination of binary variables ( $\lambda_j^{kl}$ ) and continuous design variables ( $x_j^{kl}$ ) in the CP<sup>3</sup> model presents a classical mixed-integer problem. The presence of a high number of binary commonality variables ( $\lambda_j^{kl}$ ) in the platform planning model demands extensive computational resources. A new Platform Segregating Mapping Function (PSMF) is proposed to convert the mixed-integer problem into a tractable continuous optimization problem. Unlike the VSMF mapping the PSMF mapping avoids the “all or none” assumption. Particle Swarm Optimization (PSO) was subsequently implemented by Chowdhury et al. (2011) to solve the continuous optimization problem.

### 12.3.2.1 Platform Segregating Mapping Function (PSMF)

Prior to the investigation of this new solution methodology, the commonality constraint (from Eq. 12.8) is reformulated as

$$X^T \Lambda X \leq \varepsilon \quad (12.24)$$

where  $\varepsilon$  is the aggregate tolerance specified to allow for platform creation. A careful analysis of the modified commonality constraint (Eq. 12.24) indicates that, for any two products  $k$  and  $l$ , a commonality matrix variable ( $\lambda_j^{kl}$ ) is a decreasing function of the squares of the corresponding design variable difference:  $\Delta x_j^{kl} = |x_j^k - x_j^l|$ . At the same time, the design variable differences between various pairs of products are often not independent of each other. A function that can represent the commonality between two products ( $k$  and  $l$ ) with respect to a particular design variable ( $j$ ) is similar to the function that relates the commonality variable,  $\lambda_j^{kl}$ , to the design variables  $x_j^k$  and  $x_j^l$ . This function must have the following properties:

1. The function must be continuous (defined at  $\Delta x_j^{kl} = 0$ ) and well behaved.
2. The function must have a maximum at  $\Delta x_j^{kl} = 0$  and must then decrease when  $\Delta x_j^{kl}$  increases (asymptotically tending to zero).
3. These functions, collectively, must allow a coherent consideration of the commonalities between various pairs of products with respect to a particular design variable.

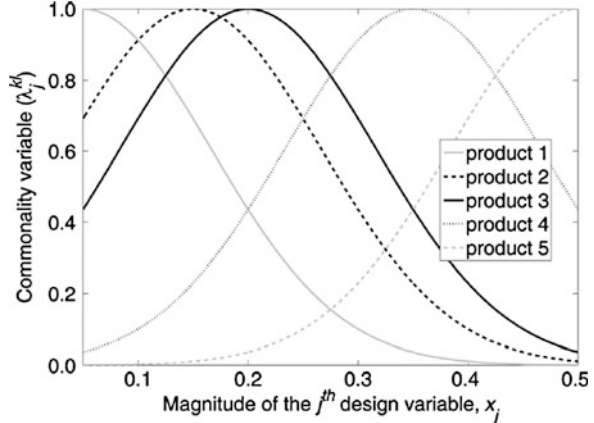
A set of Gaussian distribution functions, collectively called the Platform Segregating Mapping Function (PSMF), is developed to approximate the relationship between the commonality variables and the corresponding physical design variables. The required properties listed above are inherent in this set of Gaussian distribution functions. Interestingly, each Gaussian function in the set also provides a measure of the probability ( $p_{kl}^j$ ) of the  $j$ th design variable to be shared between product- $k$  and product- $l$ . It is helpful to note that other functions that have similar properties can also be implemented to construct the PSMF. The PSMF that relates product- $k$  and product- $l$  with respect to the  $j$ th design variable is given by

$$\lambda_j^{kl} = a \exp \left( - \frac{(x_j^k - x_j^l)^2}{2\sigma_j^2} \right) \quad (12.25)$$

where the coefficient  $a$  is assumed to be equal to 1. In Eq. (12.25), the design variable value ( $x_j^l$ ) serves as the mean of the Gaussian kernel, and the parameter  $\sigma_j$  represents the standard deviation of the Gaussian kernel for the  $j$ th design variable. This standard deviation can be determined by specifying the *full width at one-tenth maximum* ( $\Delta x_{10}$ ) as expressed by



**Fig. 12.5** Platform segregating mapping function (PSMF) for a sample 5-product family (© ASME 2011), reprinted with permission



$$\sigma = \frac{\Delta x_{10}}{2\sqrt{2 \ln 10}}, \quad \text{where } p\left(x_j^l \pm \Delta x_{10}\right) = \frac{1}{10} \tag{12.26}$$

where  $p(x)$  represents the probability at  $x$ . Subsequently, the commonality matrix can be expressed as

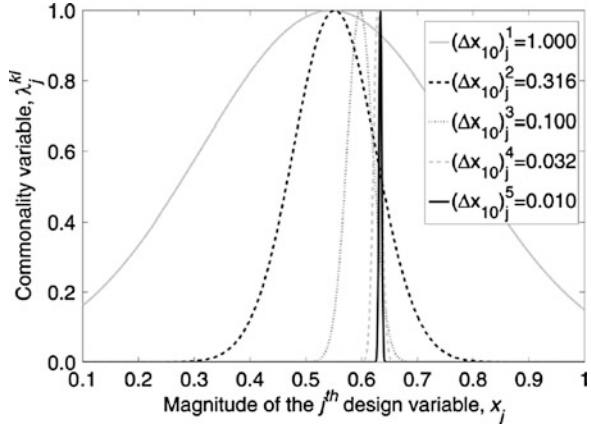
$$\lambda = \text{PSMF}(X) \tag{12.27}$$

A representative plot of the PSMF for a particular design variable ( $j$ th variable, normalized), in a sample family of five products, is shown in Fig. 12.5. In this figure, the design variable for product- $k$ ,  $x_j^k$ , can be mapped onto the Gaussian kernel of any other product- $l$  (where  $l \neq k$ ), yielding the corresponding commonality variable,  $\lambda_j^{kl}$ . By providing a pairwise representation of commonality among products, the PSMF mapping allows a product platform to be formed whenever a design variable value is the same across more than one product. It can be readily observed that this set of distribution functions (PSMF) provides a unique representation of the product commonalities by converting an essentially combinatorial problem into a tractable continuous problem. This representation can also be used to further investigate the underlying mathematics of product platform planning.

In the CP<sup>3</sup> optimization, initially, an optimal design that maximizes performance is obtained separately for each product variant in the family. The optimized design variable values, thus determined, are used to set a modified range,  $\Delta x_j$ , for the application of the PSMF on each design variable ( $j$ th design variable), similar to the approach in the SIO technique (Khire and Messac 2008). This modified range is used to evaluate the full width at one-tenth maximum ( $(\Delta x_{10})_j$ ) for the  $j$ th design variable, using

$$(\Delta x_{10})_j = \overline{\Delta x}_{10} \times \Delta x_j, \quad \overline{\Delta x}_{10} \in [0, 1] \tag{12.28}$$

**Fig. 12.6** Five consecutive stages of PSMF application for the commonality between two products ( $k$  and  $l$ ) with respect to the  $j$ th design variable



where  $\overline{\Delta x}_{10}$  represents the *normalized full width at one-tenth maximum*, which is explicitly specified during the execution of the algorithm.

The CP<sup>3</sup> model is solved using a sequence of  $N_{\text{stage}}$  Particle Swarm Optimizations (PSOs), with decreasing values (in a geometric progression) of the parameter  $\overline{\Delta x}_{10}$ . This multistage optimization results in sharper Gaussian kernels with every subsequent stage, rendering a progressively rigorous application of the commonality constraint; this process is illustrated in Fig. 12.6. This figure shows a five-stage application of the PSMF, with the specified initial and final normalized full width at one-tenth maximum ( $\overline{\Delta x}_{10}^{\text{max}} = 1.0$  and  $\overline{\Delta x}_{10}^{\text{min}} = 0.1$ ). In the final stage, design variable values (e.g.,  $x_j^k$  and  $x_j^l$ ) residing within the same Gaussian kernel (sharp dome in Fig. 12.6) would practically indicate that the corresponding products (product- $k$  and product- $l$ ) share the  $j$ th design variable.

Optimization is performed on the approximated MINLP problem to minimize a simplified weighted sum of the performance and the cost objectives. The modified optimization problem is defined as

$$\text{Max } w_1 f_{\text{per}}(X) + (1 - w_1)(-f_{\text{cost}}(X))$$

subject to

$$X^T \Lambda X \leq \epsilon$$

$$g_i(X) \leq 0, \quad i = 1, 2, \dots, p$$

$$h_i(X) = 0, \quad i = 1, 2, \dots, q$$

$$\Lambda = f_{\text{con}}(\lambda)$$

$$\lambda = \text{PSMF}(X)$$

$$X = [x_1^1 \quad x_1^2 \quad \dots \quad x_1^N \quad \dots \quad x_j^1 \quad x_j^2 \quad \dots \quad x_j^N \quad \dots \quad x_n^1 \quad x_n^2 \quad \dots \quad x_n^N]^T$$

$$\lambda_j^{kl} \in B : B = \{0, 1\}$$

(12.29)

---



---

Step 1	: Optimize each product using PSO (maximizing performance)
Step 2	: Determine the range for implementing PSO on each $x_j$
Step 3	: Initiate a random population of size $N_{pop}$
Step 4	: Set $\overline{\Delta x}_{10} = \overline{\Delta x}_{10}^{\max}$ and $i_{stage} = 1$
Step 5	: Simultaneously optimize $N$ products using PSO (solve Eq. 21)
Step 6	: Set $\overline{\Delta x}_{10}^{i_{stage}+1} = \overline{\Delta x}_{10}^{i_{stage}} \times \overline{\Delta x}_{10}^{frac}$ , where $\Delta x_{10}^{frac} = \left( \frac{\overline{\Delta x}_{10}^{\min}}{\overline{\Delta x}_{10}^{\max}} \right)^{\frac{1}{N_{stage}-1}}$
Step 7	: Evaluate $(\Delta x_{10})_j^{i_{stage}+1}$ using Eq. 20
Step 8	: Choose the optimal configuration as one of the starting points
Step 9	: Initiate a random population of size $N_{pop} - 1$ , and set $i_{stage} = i_{stage} + 1$
Step 10	: If $i_{stage} < N_{stage}$ go to Step 5, else terminate solution

---



---

**Fig. 12.7** Pseudocode for the application of PSMF using the PSO algorithm

In this problem definition, a value of 0.5 is used for the objective weight  $w_1$ , and the term  $PSMF(X)$  is given by Eq. (12.25). The objective  $f_{per}$  is given by Eq. (12.4), and the objective  $f_{cost}$  is evaluated using a cost decay function. This cost decay function accounts for the volume of production; details of the cost decay function can be found in the paper by Chowdhury et al. (2011). As is well known, the weighted-sum method entails certain limitations associated with non-convex Pareto frontiers. Importantly, we note that the  $CP^3$  optimization approach can be implemented using other powerful methods that aggregate multiple objectives into one objective such as Physical Programming (Messac et al. 2002a, b). The process of application of the PSMF technique using PSO is summarized by the pseudocode in Fig. 12.7.

### 12.3.2.2 Choice of Optimization Algorithm

The optimization process in the  $CP^3$  method is performed using an effective variation of the standard Particle Swarm Optimization (PSO) algorithm. Particle Swarm Optimization is one of the most popular heuristic optimization algorithms, introduced by Kennedy and Eberhart (1995). Conceptually, the search characteristics of PSO mimic the natural collective movement of animals, such as bird flocks, bee swarms, and fish schools. In the recent literature, PSO has been used to address various aspects of product family design (Yadav et al. 2008; Moon et al. 2011).

It is however important to note that neither the optimization of the  $CP^3$  model nor the application of the PSMF-based solution strategy is restricted to the use of the PSO algorithm. Owing to the generic formulation of the PFD process provided by the  $CP^3$  model, the approximated continuous optimization problem obtained using the PSMF technique can also be solved using other standard algorithms, such as the sequential quadratic programming (SQP), the real-coded NSGA-II

algorithm (Deb et al. 2002), the strength Pareto evolutionary algorithm (SPEA: Zitzler et al. 2004), the differential evolution algorithm (Price et al. 2005), and the single-objective modified predator–prey (MPP) algorithm (Chowdhury and Dulikravich 2010).

Population-based heuristic algorithms are however preferable in this case, since the commonality constraint is expected to be multimodal. Heuristic algorithms, such as NSGA-II (Deb et al. 2002) and MPP (Chowdhury et al. 2009), are typically useful for multi-objective optimization; these algorithms can be leveraged to explore a bi-objective optimization scenario with performance and cost as separate objectives. The original MINLP problem yielded by the CP<sup>3</sup> model can also be directly solved using typical MINLP solvers such as branch and bound and cutting plane techniques or using binary genetic algorithms (e.g., bin-NSGA-II). However, solving the MINLP problem directly may prove to be unreasonably challenging and computationally expensive owing to the high dimensionality of the commonality variables.

## 12.4 Application to Family of Electric Motors

### 12.4.1 Application of the SIO Method

The SIO method was applied by Khire (2006) to design a family of ten electric motors that satisfies ten different torque requirements, as specified in the book-chapter by Simpson (2006). Sequential quadratic programming (SQP) was used to perform the optimization. The SIO method converged to a product family design in which the variables  $A_{wa}$ ,  $A_{wf}$ ,  $N_s$ ,  $N_c$ ,  $L$ , and  $t$  were shared by all the motors (i.e., platform variables) and the variable  $r_0$  scaled across the family (scaling variable). The design variable values of the ten motors in the optimized family are provided in Table 12.3 [reported by Khire (2006)]. In this table, the  $i$ th motor is depicted by  $M-i$ ; the variables  $A_{wf}$  and  $A_{wf}$  are expressed in mm<sup>2</sup>; the variables  $L$ ,  $r_0$ , and  $t$  are expressed in cm; and the variable  $I$  is expressed in Amp.

The overall platform plan of the motors obtained by the SIO method is similar to that obtained using the two-stage method [described in Sect. 3.3 of the book-chapter by Simpson (2006)]. Motor designers have confirmed that varying the stator radius is indeed one of the most effective ways of achieving torque variations among motor variants. However, varying stack length is generally more cost effective from a manufacturing perspective according to Black and Decker [Sect. 3.3 of the book-chapter by Simpson (2006)].

Khire (2006) also provided an investigation of the robustness of the SIO methodology; application of the method was analyzed for different starting values of the VSMF parameter,  $\alpha$ , and for different values of its specified increment between iterations ( $\Delta\alpha$ ). Khire (2006) concluded that in order to save computation time, it is advisable to start at  $\alpha = 0.9$  and terminate the iteration once the trend of

**Table 12.3** Design variable values of the optimized motor family (obtained by SIO) (Khire 2006)

Var.	M-1	M-2	M-3	M-4	M-5	M-6	M-7	M-8	M-9	M-10
$N_c$	1,500	1,500	1,500	1,500	1,500	1,500	1,500	1,500	1,500	1,500
$N_s$	165.1	165.1	165.1	165.1	165.1	165.1	165.1	165.1	165.1	165.1
$A_{wvf}$	0.141	0.141	0.141	0.141	0.141	0.141	0.141	0.141	0.141	0.141
$A_{wa}$	0.362	0.362	0.362	0.362	0.362	0.362	0.362	0.362	0.362	0.362
$r_0$	4.574	4.734	4.809	4.879	5.008	5.123	5.225	5.315	5.394	5.523
$t$	0.433	0.433	0.433	0.433	0.433	0.433	0.433	0.433	0.433	0.433
$L$	2.06	2.06	2.06	2.06	2.06	2.06	2.06	2.06	2.06	2.06
$I$	3.119	3.21	3.257	3.304	3.401	3.499	3.599	3.701	3.805	4.018
$\eta$	0.836	0.812	0.801	0.789	0.767	0.745	0.725	0.705	0.686	0.649

design variable segregation is observed. The value of  $\Delta\alpha$  must be selected such that it fulfills two criteria: (1) convergence of the solution and (2) manageability of the computation cost. Based on numerical experimentation, a favorable range for the  $\Delta\alpha$  value between 0.06 and 0.1 was suggested by Khire (2006). It is helpful to note that the suggestions regarding the values of initial  $\alpha$  and  $\Delta\alpha$  (presented here) are particularly applicable for the family of motors and hence are not necessarily universal.

### 12.4.2 Application of the CP<sup>3</sup> Method

The CP<sup>3</sup> method was applied by Chowdhury et al. (2011) to design the same family of ten electric motors (with ten different torque requirements) as specified in the book-chapter by Simpson (2006). Initially, optimization is performed on each motor individually (using PSO), in order to maximize the individual motor efficiencies subjected only to the physical design constraints (specified in Eq. 12.3). A new set of variable limits is determined from the highest and the lowest values of each optimized design variable, among the ten motors. The new design variable limits are used to execute step 4 to step 9 of the pseudocode given in Fig. 12.7.

The optimized platform plan obtained for the motor family is illustrated in Table 12.4. Each uppercase letter in Table 12.4 represents a platform; blocks in Table 12.4, displaying similar letters, imply that the corresponding products are members of a particular platform (that share the corresponding design variable). A block displaying the “—” symbol represents a scaling design variable value, thereby implying that the corresponding design variable value is not shared by more than one product. The efficiencies of the motors in the optimized family are also provided in Table 12.4.

Ten sub-platforms are formed in the motor family. The design of Motor-10 is observed to be completely unique (no variable sharing with other motors). On the other hand, Motor-3 shares the maximum number of variables with one or more other motors. None of the design variables are shared across all the products. The thickness of the stator exhibits a strong platform property—Motor-1 and

**Table 12.4** Optimized platform plan of universal electric motors (obtained by CP<sup>3</sup>) (© ASME 2011), reprinted with permission

Var.	M-1	M-2	M-3	M-4	M-5	M-6	M-7	M-8	M-9	M-10
$N_c$	–	–	–	–	–	–	A	–	A	–
$N_s$	–	–	–	–	–	–	–	–	–	–
$A_{wf}$	C	D	C	C	D	C	–	E	E	–
$A_{wa}$	–	–	B	–	–	–	B	–	–	–
$r_0$	F	G	F	–	–	–	G	–	–	–
$t$	H	I	H	I	I	I	I	I	I	–
$L$	–	J	J	–	–	–	–	J	–	–
$\eta$	0.875	0.857	0.834	0.898	0.901	0.864	0.831	0.657	0.727	0.731

Motor-3 have the same stator thickness, and Motor-2 and Motor-4 to Motor-9 have the same stator thickness. The number of wire turns on each field pole is different for each motor, making it a typical scaling variable.

### 12.4.3 Comparison of CP<sup>3</sup> and SIO Results

The optimal motor family produced by SIO offered significantly higher degree of commonality than that produced by CP<sup>3</sup>. On the other hand, the efficiencies of the ten individual motors are higher in the case of the family yielded by CP<sup>3</sup>. It is helpful to note that the SIO and the CP<sup>3</sup> involve different commonality objectives. Hence, a direct comparison of the performances of these two methods is not feasible.

A comparison of their results can however be made if the commonality in the final optimized motor families is represented in terms of the standard commonality index [CI (Martin and Ishii 1996)], which is mathematically defined as

$$CI = 1 - \frac{u - \max(n_k)}{\sum_{k=1}^N n_k - \max(n_k)} \quad (12.30)$$

where  $u$  represents the actual number of unique parts in the entire product family and  $n_k$  represents the number of parts in the  $k$ th product. The “ $-\max(n_k)$ ” term is included in the definition to ensure that the CI varies between 0 and 1. According to this definition:

- When the product variants in a family are identical (i.e., all parts are shared among all products), the value of CI is a maximum of 1.
- When the product variants are completely different from each other (i.e., no parts are shared among the product variants), the value of CI is a minimum of zero.

**Table 12.5** Comparison of the optimized motor families obtained by the SIO and the CP<sup>3</sup> methods

Method	Average motor efficiency	Actual number of unique parts	Commonality index (CI)
SIO	0.7515	16	6/7
CP <sup>3</sup>	0.8175	52	2/7

In the case of the motor family, each motor variant comprises seven variables that participate in platform planning, i.e.,  $n_k = 7$ . For the optimized motor family obtained by the SIO method, the actual number of unique parts ( $u$ ) is 16. The actual number of unique parts ( $u$ ) in the optimized motor family obtained by the CP<sup>3</sup> method is 52. In order to facilitate easy comparison, the results of the optimized motor families obtained by the two methods are summarized in Table 12.5.

It is observed from Table 12.5 that the average motor efficiency of the optimized family obtained by CP<sup>3</sup> is approximately 9 % higher than that obtained by SIO. On the other hand, the commonality (in terms of CI) in the optimized motor family obtained by SIO is three times of that obtained by CP<sup>3</sup>. Hence, the optimized motor families obtained by the two methods are trade-off solutions with respect to each other. It is also important to note that both methods use an aggregate objective function, in which the performance objective has a weight of 0.5 (the second objective being different); if lower weights are used for the performance objective, the resulting total number of unique parts (in the optimized family) is expected to decrease in both methods.

Overall, it is evident that since the CP<sup>3</sup> method is not restricted by the “*all or none*” assumption (as seen from the results in Table 12.4), it is applicable to a wider variety of commercial product families (compared to SIO). An “*all or none*” approach would also demand a higher compromise of the product performances to achieve commonality among products, which was the case when the SIO method was applied. However, the optimization involved in the SIO method is more tractable than that involved in the CP<sup>3</sup> method, since the latter yields multimodal commonality constraints; this attribute makes SIO relatively easier to implement. Therefore SIO can be a particularly useful product family design method in cases where the user desires to have an “*all or none*” platform plan.

## 12.5 Closing Remarks

Quantitative design of product families should be an integral part of the entire *concept-to-shelf* process for a line of products. Unfortunately, the planning of product platforms and the quantification of the individual product attributes are often a complex and system-dependent mathematical problem—involving a mix of integer and continuous variables and highly nonlinear functions. Simplification of the product platform planning process and the development of generic protocols

are therefore necessary to take the quantitative PFD approaches from research labs to industrial applications.

This chapter summarizes the formulation of two methods that seek to develop and solve a more tractable continuous approximation of the PFD problem. The Selection-Integrated Optimization (SIO) method uses a continuous mapping function to quantify the tendency of a variable to become common among products. An optimization-based iterative process is used to simultaneously segregate the design variables into platform and scaling types and quantify the optimal variable values. Similarly, the Comprehensive Product Platform Planning (CP<sup>3</sup>) method uses a set of continuous kernel functions to map each design variable's tendency towards commonality for each pair of products (in the family). Together with the mapping scheme, Particle Swarm Optimization is implemented through an iterative process to segregate the design variables into platform and scaling types. The pairwise mapping strategy allows the CP<sup>3</sup> method to avoid the “*all or none*” assumption. More importantly, the CP<sup>3</sup> method formulates a generic *product platform planning* model—one that can be solved using any standard continuous optimization method.

The conversion of a complex PFD problem into a tractable continuous form helps the designer better exploit the potential of quantitative optimization. We encourage interested researchers to build on the foundation laid down by these two PFD methods and develop more comprehensive (yet tractable) one-step approaches that can be readily applied to a wider variety of problems.

**Acknowledgements** This work has been supported by the National Science Foundation under Awards no. CMMI 0946765 and CMMI 1100948. Any opinions, findings, conclusions, and recommendations presented in this chapter are those of the authors and do not reflect the views of the National Science Foundation.

## References

- Chen C, Wang LA (2008) Modified genetic algorithm for product family optimization with platform specified by information theoretical approach. *J Shanghai Jiaotong Univ (Sci)* 13:304–311
- Chowdhury S, Dulikravich GS (2010) Improvements to single-objective constrained predator–prey evolutionary optimization algorithm. *Struct Multidiscip Optim* 41:541–554
- Chowdhury S, Dulikravich GS, Moral RJ (2009) Modified predator–prey algorithm for constrained and unconstrained multi-objective optimisation. *Int J Math Model Numer Optim* 1:1–38
- Chowdhury S, Messac A, Khire R (2011) Comprehensive product platform planning framework. *ASME J Mech Des* (special issue on Designing Complex Engineered Systems) 133:101004-1–101004-15
- Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: Nsga-II. *IEEE Trans Evol Comput* 6:182–197
- Fellini R, Kokkolaras M, Papalambros PY, Perez-Duarte A (2005) Platform selection under performance bounds in optimal design of product families. *ASME J Mech Des* 127:524–535
- Jiao J, Zhang Y, Wang Y (2007a) A generic genetic algorithm for product family design. *J Intell Manuf* 18:233–237



- Jiao J, Simpson TW, Siddique Z (2007b) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18:5–29
- Kennedy J, Eberhart RC (1995) Particle swarm optimization. *IEEE Int Conf Neural Netw* 4:1942–1948
- Khajavirad A, Michalek JJ (2008) A decomposed gradient-based approach for generalized platform selection and variant design in product family optimization. *ASME J Mech Des* 130:071101-1–071101-8
- Khajavirad A, Michalek JJ, Simpson TW (2009) An efficient decomposed multiobjective genetic algorithm for solving the joint product platform selection and product family design problem with generalized commonality. *Struct Multidiscip Optim* 39:187–201
- Khire R (2006) Selection-integrated optimization (SIO) methodology for adaptive systems and product family optimization. PhD Thesis, Rensselaer Polytechnic Institute, Troy, NY
- Khire R, Messac A (2008) Selection-integrated optimization (sio) methodology for optimal design of adaptive systems. *ASME J Mech Des* 130:101401-1–101401-13
- Khire R, Messac A, Simpson TW (2006) Optimal design of product families using selection integrated optimization (SIO) Methodology. In: 11th AIAA/ISSMO multidisciplinary analysis and optimization conference, AIAA-2006-6924, Portsmouth, VA, September
- Martin M, Ishii K (1996) Design for variety: a methodology for understanding the costs of product proliferation. In: ASME design engineering technical conferences and computers in engineering conference, ASME, Irvine, CA, 96-DETC/DTM-1610
- Messac A, Martinez MP, Simpson TW (2002a) Effective product family design using physical programming. *Eng Optim* 124:245–261
- Messac A, Martinez MP, Simpson TW (2002b) Introduction of a product family penalty function using physical programming. *ASME J Mech Des* 124:164–172
- Moon SK, Park KJ, Simpson TW (2011) Platform strategy for product family design using particle swarm optimization. In: ASME 2011 international design engineering technical conferences, Washington, DC
- Price K, Storn RM, Lampinen JA (2005) *Differential evolution: a practical approach to global optimization*, 1st edn. Springer, New York
- Simpson TW (2006) Methods for optimizing product platforms and product families. In: Simpson TW, Siddique Z, Jiao J (eds) *Product platform and product family design*. Springer, New York, pp 133–156
- Simpson TW, Siddique Z, Jiao RJ (2006) Platform-based product family development. In: Simpson TW, Siddique Z, Jiao RJ (eds) *Product platform and product family design*. Springer, New York, pp 1–15
- Watson LT, Haftka RT (1989) Modern homotopy methods in optimization. *Comput Meth Appl Mech Eng* 74:289–305
- Yadav SR, Dashora Y, Shankar R, Chen FTS, Tiwari MK (2008) An interactive particle swarm optimisation for selecting a product family and designing its supply chain. *Int J Comput Appl Technol* 31:168–186
- Zitzler E, Laumanns M, Bleuler S (2004) A tutorial on evolutionary multiobjective optimization. In: *Metaheuristics for multiobjective optimisation*. Springer, Berlin, pp 3–37

# Chapter 13

## Defining Modules for Platforms: An Overview of the Architecting Process

Katja Hölttä-Otto, Kevin N. Otto, and Timothy W. Simpson

**Abstract** Product platforms have shown to provide significant cost and time savings while still allowing companies to offer a variety of products. As a result, a multitude of methods have been developed to design product platforms. These methods, however, have been developed independent of one another, and it can be daunting to try to compare the methods and understand which approach might be suitable when or how the methods might interlink, if at all. In this chapter, we review the platform architecting process and tie together several approaches introduced both in this book and the existing literature. A family of unmanned ground vehicles (UGVs) is used as an illustrative example to demonstrate several of these approaches and their integration.

### 13.1 Introduction

A product platform is a collection of common assets that are shared across a product family, often over product generations. These assets can be anything from a manufacturing line to common system models. A modular platform is a type of platform where the common shared assets are modules. Modules, on the other hand, are typically subassemblies of products that have an easily identifiable function.

---

K. Hölttä-Otto (✉)

Engineering Product Development, Singapore University of Technology and Design,  
Singapore 138682, Singapore  
e-mail: [Katja\\_Otto@sutd.edu.sg](mailto:Katja_Otto@sutd.edu.sg)

K.N. Otto

Singapore University of Technology and Design, Singapore, Singapore

T.W. Simpson

Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA  
Industrial and Manufacturing Engineering, Penn State University, University Park,  
PA 16802, USA

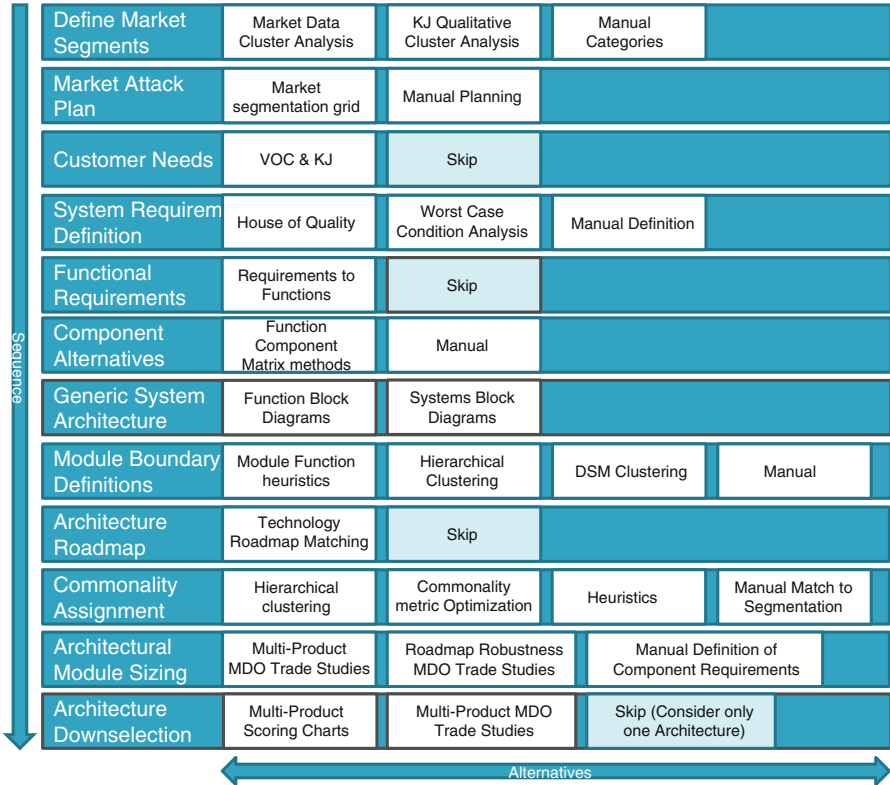


Fig. 13.1 Architecting steps and possible methods and techniques

Such modules are ideally tightly connected within the module but loosely connected to the rest of the product.

Product platforming has become a common approach to develop product families, and generations of products, in a cost-effective manner. The sharing of assets across the platform products brings cost savings and many operational benefits in manufacturing, logistics, and quality control, for example. Furthermore, since each product is based on a set of common modules, developing additional variants or a new version of past products is easier, since only the variant or updated module will need to be designed.

Over the years there has been active work in developing methods to define these modules for product families: methods to map requirements of a family to a set of products; methods to define the common or unique platform modules; methods to optimize variety, cost, commonality, or other parameters; methods to evaluate platforms; etc. Figure 13.1 shows multiple types of methods and how they can be situated in the overall platform development process from defining customer needs and market segments down to architectural module sizing or architectural down selection. Each method has typically been developed independently of other methods, and it is not clear if and how these methods could be used jointly.

In this chapter we link multiple methods discussed in this book and in the broader literature into a logical, structured process. We do this by presenting a step-by-step approach to transition requirements to a final platform using and referring to relevant methods along the way. The goal in this chapter is to show how the various alternative approaches and methods to platforming can contribute to the overall goal of developing a successful product family.

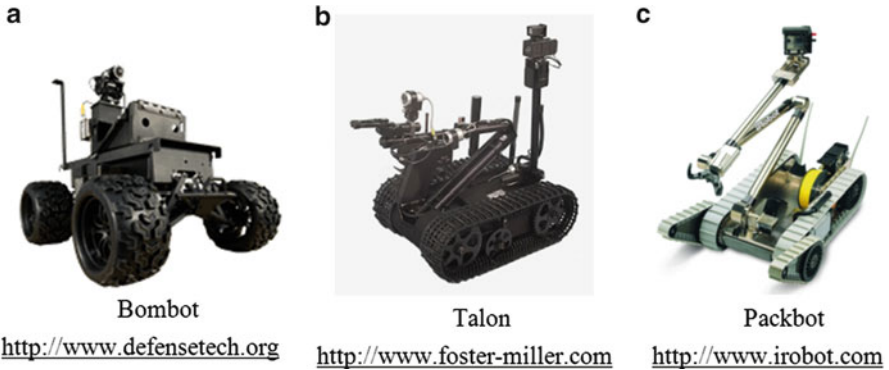
The rest of this chapter discusses a platform development process in the order shown in Fig. 13.1. This is a prescription for a logical sequence of steps to apply in order, with multiple alternatives at each step as indicated in Fig. 13.1. We do not assert this is the only sequence one can take through these steps; in general, one can find design problems where alternative orders are appropriate. However, Fig. 13.1 is logical and a useful starting point for any firm or stakeholder new to the platform-based development. In each of the next sections, we discuss the various methods suitable for the steps as well as show an illustrative example. We end with a chapter summary and list of implications for platform design.

## 13.2 Market Segment Definition and Market Attack Plan

The start of a product development effort should entail defining the population of customers and applications for the product(s). Typically this starts with an observation of a perceived need in an application domain and extension into exploring the range of different market geographies and demographics that have potential similarities in needs. Market segmentation helps identify potential clusters of customers with similar needs, which enables designing products for each market segment, rather than one product to meet all the vastly different needs.

Initially, one can define a wide range of characteristics upon which to subdivide a population of potential customers, such as applications, geographic boundaries, and demographics. However, one will also find that many such distinctions are artificial and that there is no real difference in customer needs between several over-partitioned divisions of the population. For each smallest partition of the population, one can conduct surveys of customer demographics, use applications, or even customer needs analysis to establish clear distinctions amongst the population. Clustering these results into internally homogeneous and externally heterogeneous groups helps form clear *market segments*.

The simplest approach to identifying these clusters of similar customers is to manually cluster on characteristics such as use or business application as market segments. For example, defense-related applications form a different segment than commercial applications, and geographic/regional variations may lead to distinct segmentation. Other options could be use of standard clustering methods, such as hierarchical clustering, or more advanced methods such as fuzzy clustering (Moon et al. 2006; Zhang et al. 2007). While we recommend more rigorous approaches in both this and further steps, a simpler and quicker method is sometimes desired. Regardless of the clustering method chosen, no differences in results can indicate



**Fig. 13.2** Examples of existing unmanned ground vehicles

over-partitioning of the market. Unfortunately this is not always obvious, and establishing differences in markets and targeted products is a profitable area of study.

Throughout this chapter, a family of *unmanned ground vehicles* (UGVs) for explosive ordnance disposal is used as an example. UGVs have civilian and military applications, and many systems exist in the market (see Fig. 13.2). Smaller UGVs may provide one-time use to detonate an explosive remotely once it has been located in the field (i.e., the UGV may be destroyed along with the ordnance) while larger UGVs may perform additional functions such as sensing, defusing, or disposing the ordnance, allowing for repeated use and multiple missions.

The primary benefit of UGVs is reducing casualty risk of ordnance disposal regardless of whether it is civilian or military; however, there are many potential applications for UGVs. Partitioning these potential applications can lead to a form of market segmentation. For example, one could partition the UGV user population into the following applications:

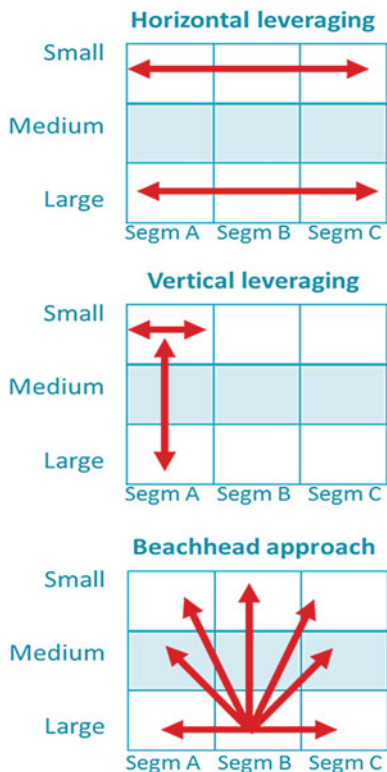
1. Explosive abatement—ordnance detection, disabling, and disposal
2. Hazardous sites—hazard sensing and locating in unhealthy environments
3. Combat—providing attack capability for high-risk missions
4. Reconnaissance—providing site observations and intelligence
5. Target and decoy—providing a target that simulates an enemy vehicle
6. Civil and commercial—UGVs for commercial transport

Meanwhile, the range and autonomy of the UGVs can provide a second axis to categorize potential customer segments:

1. Micro, line of sight
2. Small remote operated, 2-mi. range
3. Tactical remote operated, 50-mi. range
4. Long endurance autonomous, 100-mi. range

The resulting matrix of these two categories (application versus range) defines a set of different potential market segments for UGVs. This follows the product

**Fig. 13.3** Platform leveraging strategies [adapted from Meyer and Lehnerd (2000)]

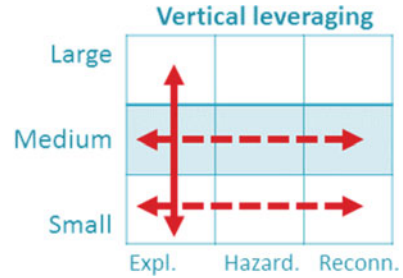


market matrix approach advocated by Kotler and Keller (2009) and Meyer and Lehnerd (2000). In their approach the rows of the matrices correspond to market segments and the columns to product segments, which helps designers identify how platform components may be leveraged across multiple segments (see Fig. 13.3). Several of these segments may be identical in terms of specific customer needs; however, this can be determined later when customer interviews and analyses are conducted.

Given a market segmentation, a decision must be made about how to “attack” the market over time. The corresponding *market attack plan* determines whether new products are offered simultaneously to all segments in parallel or rolled out sequentially over time to different market segments. Key considerations are the addressable market size of each segment and the degree of difference among the segments [see Chaps. 2 and 3 in Simpson et al. (2005)].

To develop the market attack plan for our UGV product family example, we consider the weight, speed, range, and lift capacity requirements for different potential segments within a *product market matrix* defined by applications versus range. Over 50 different potential applications were identified in the matrix based on type of ordnance, functionality (e.g., dig, detonate, diffuse), location of operation, etc. (Simpson et al. 2012). The UGV market segmentation matrix was then

**Fig. 13.4** Planned market strategy for the UGVs



manually clustered into three homogeneous segments to be consistent with current systems in the market. Three “performance tiers” were identified corresponding to small, medium, and large UGVs based on weight. Therefore, the columns are segmented into small, medium, and large UGVs in the market segmentation matrix for the UGV example, and the rows are divided into explosion abatement, hazardous sites, and reconnaissance applications.

Based on this segmentation, a vertical leveraging strategy may be possible as illustrated by the solid vertical arrow in Fig. 13.4. This would enable a new family of UGVs to be developed and scaled up for this initial segment, with platform modules created to satisfy requirements for the small, medium, and large UGVs. In the future, the platform could be extended by leveraging these modules horizontally to other market segments as shown by the dashed arrows in Fig. 13.4.

We note that one can often associate vertical leveraging to *scalable platforms* where different sizes of the same modules and components are used. Meanwhile, horizontal leveraging can be generally associated with *swappable modules* where a common subset of core modules is reused across different products that are differentiated by integrating the core with swappable unique modules thereby providing different functionality. Successful platforms often utilize a combination of scaling and modularity to attack the different market segments in a strategic and cost-effective manner.

### 13.3 Customer Needs Gathering

Given the list of market segments and a strategy for attack based on the addressable market and their differences, a set of testable, measurable requirements is needed to “design to.” The requirements must be based on what the customer seeks in the product. In platform development project, the objective is to create several products for different segments, each with different needs that drive the individual product differences. Therefore, customer needs may be gathered for each individual product separately. In other words the intended variety in the product family is designed first before defining what the common modules in a modular platform should be.

There are well-established methods to gather customer needs such as *voice of the customer* (VOC) (Griffin and Hauser 1993; Churchill and Iacobucci 2004).

Here, interviews are conducted with randomly sampled people from each market segment and questioned over how they use the product. This elicits qualitative and quantitative need statements which they seek from the product.

For the UGV example, the customer requirements were gathered from multiple requirements working groups that met over a period of about 2 years. Each group was comprised of representatives from different branches of the military and included senior personnel, ordnance disposal experts, and technicians involved with logistics, maintenance, and support.

For each “segment” in the market segmentation grid in Fig. 13.4, requirements were defined (e.g., weight, range, speed, manipulator length), and threshold and objective values were identified for each requirement. The threshold value is the minimum value that must be met in order to satisfy a requirement while the objective value provides a target that users would like to achieve. Threshold and objective values were defined for the small, medium, and large UGVs, which were used to define the system requirements as discussed next.

### 13.4 System Requirements Definition

Once the customer needs for each market segment and corresponding product variant are clearly defined, the next step is to define quantitative verifiable system requirements for each product variant. Again, there are well-established methods for this step, including the House of Quality (Hauser and Clausing 1988). Another option is to define the worst-case operating conditions and define the system specifications for those cases. Perhaps the simplest option for this step is to convert the customer needs into system requirements similar to target value setting as described by Ulrich and Eppinger (2004) or requirement definition following the INCOSE guidelines (INCOSE 2010). Regardless of what method is used, at the end of this step, quantified system requirements have been identified to which provide design targets to ensure each variant is capable of satisfying the customer needs in its corresponding market segment.

Table 13.1 gives an example of the system requirements that were defined for the UGV example following the customer needs analysis. Threshold and objective values for each requirement are defined for each weight class. This information can be used to help identify common, variant, and unique requirements, i.e., those that are the same (identical) among all three UGVs, those that are similar but vary slightly from one UGV to the next, and those that are unique to a specific UGV. For example, some of the maneuvering, sensing, and communication requirements are the same for each UGV; however, the range and payload requirements vary for each weight class. Finally, the manipulator, reach, drag/roll/push, and large object requirements are unique to each UGV, with no requirements defined for the small UGV when that capability is not present (e.g., large object pickup).

Once a specification list is completed for each product variant, it is a key milestone as it defines what the engineers should design to. Therefore, it should be



**Table 13.1** System requirement areas and corresponding UGV activities and functions

Requirement	Activities	Functions
Range (ft)	Travel to desired location	Provide propulsion
	Travel home	Store power
Slope climb (°)	Travel to desired location	Support weight
	Travel home	Support loads (Constraint)
Maneuver width (in.)		
On-board volume (in <sup>3</sup> )	Carry payload	Store payload
On-board weight (lb)		Support loads
Drag/roll/push (lb)	Move object	Provide propulsion
	Move obstacle	
Horizontal reach (in.)	Reach for object	Allow DoF (manip)
Vertical high reach (in.)	Reach for object	Allow DoF (manip)
Sensing type	Observe surroundings	Sense environment
Video vert. high reach (in.)	Capture object	Allow DoF (location)
Video horiz. reach (°)	Capture object	Allow DoF (orientation)
Large obj pickup length (in.)	Grasp object	Couple sample
Large obj pickup width (in.)	Grasp object	Couple sample
Large obj pickup height (in.)	Grasp object	Couple sample
Lift capacity (lb)	List object	Support loads
Tool precision	Use tool	
Tool size (in <sup>3</sup> )		(Constraint)
Tool weight (lb)		
Communication range (ft)	Follow instructions	Transmit/process inform
	Send data	Control operation Transmit commands/data Remote control operation

checked for various considerations. One important factor is completeness against the customer need list. Each customer need should be covered by an associated requirement—meeting the requirement will ensure satisfaction on the customer need. A second important consideration is conflict management among requirements. Customers will want high performance and low cost. Definition of the target cost and the minimum performance level defines how these conflicting goals are to be simultaneously met by the design team. Such conflicts in the requirements should be highlighted and tracked as risk items, particularly when they cross platform considerations. That is, some product variants are low cost and low performance and might be in the same platform as a high-cost high-performance variant. Platforms across conflicting requirements in this way can be difficult to engineer, and exclusion of one or the other variant from the platform may be warranted.

Next, a function-based approach or a component-based approach can be used to start to develop the platform architecture for the family. To define a platform, the function-based approach takes a more general view to define common functions independent of the form. The component-based approach, on the other hand, makes use of a priori knowledge, however acquired or assumed, of the most relevant components needed. If the function-based approach is used, that approach is followed by mapping function to form, i.e., defining the components as part of

embodiment design. Therefore, in the end both approaches lead to the same result though the function-based approach perhaps considers a broader set of alternatives. In this chapter we introduce and demonstrate both approaches.

### 13.5 Functional Requirements Definition

A *functional requirement* is a requirement to exhibit a certain functional behavior, i.e., to do something. A *constraint* is a nonfunctional requirement such as weight requirement. Defining functional requirements is discussed extensively in Otto and Wood (2001). In this chapter, we follow their suggested process of taking the customer needs and separating those into functions and constraints, then listing the customer activities of the desired system, using those activities to form a functional model, and finally comparing the resulting functional model with the original customer needs. This process results in understanding *what*, as opposed to *how*, the system should do to achieve the customer needs.

In the interest of space, we have listed the activities next to the corresponding customer requirements in Table 13.1. On a typical mission, a UGV would travel to a desired location, observe and record the surroundings, and collect objects or act on objects as instructed and communicated back home. The supporting functions for these basic customer activities include moving the camera and manipulators up, down, and horizontally. Since this is an existing family of robots, our choice of functions was influenced by the existing configuration of the robot family. This will result in a similarity between the embodiment of this functional model and the design created using the component-based approach. The resulting functional model is shown in Fig. 13.5. Table 13.1 shows how this functional model meets the customer requirements.

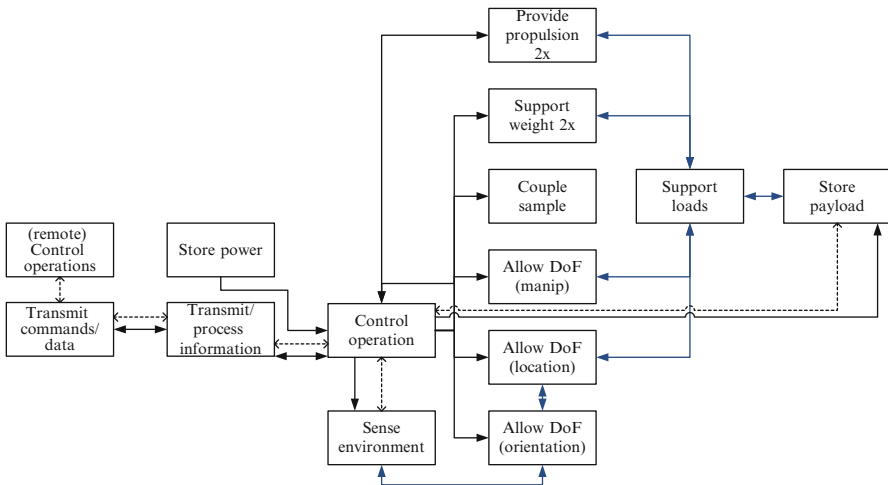


Fig. 13.5 Functional model of a UGV

### 13.6 Component-Based Approach

Components can be defined either directly from system requirements or by mapping functions, as defined in the previous step, to form part of the system embodiment phase (Pahl and Beitz 1996). There are at least two well-established methods that have incorporated the mapping of requirements to components. In Martin and Ishii’s (2000) *Design for Variety* method, they extend the House of Quality to create a function-component matrix for calculating the *Generational Variety Index* (GVI) for each subsystem. A key step in this matrix is to map the engineering requirements into subsystems, or components, in order to identify how much effort is needed to redesign a component if there was a change in a specific requirement. Subsystems with a high GVI value will undergo significant redesign in order to satisfy the range of customer needs while low GVI values indicate components that will remain relatively stable across the family. Figure 13.6 shows the GVI values for the UGV example (Simpson et al. 2012). As seen, the manipulator and chassis will vary substantially within the family (i.e., high GVI values) while cameras and OCU (Operator Control Unit) will have little variation.

Requirement	Subsystem										
	Chassis	Battery	Tracks	Communication box	Electronics box	Manipulator	Gripper	Cameras	Payload bay	Antennae	OCU
Range (feet)				6						6	
Slope Climb (deg)	3		3			1					
Maneuver width (in)	3		3								3
On board vol (in^3)	6				3				6		
On board wt (lb)	6				3				6		
Drag/Roll/Push (lb)	6	3	6			6	6				
Horiz reach (in)	6	3				9	1				
Vert high reach (in)	1					9	1				
Sensing type									3		
Video vert high reach (in)	1					9	1				
Large Obj Pickup (length)						3	6				
Large Obj Pickup (width)						3	6				
Large Obj Pickup (height)						3	6				
Lift capac (lb)	6					9	3				
Tool precision				1				6	1		1
Tool size (in^3)						3	6				
Tool wt (lb)						3	6				
Comm range (ft)		3		6						6	
<b>GVI Values</b>	<b>38</b>	<b>9</b>	<b>12</b>	<b>13</b>	<b>6</b>	<b>58</b>	<b>48</b>	<b>1</b>	<b>15</b>	<b>12</b>	<b>4</b>

Fig. 13.6 Generational variety index for UGV example (Simpson et al. 2012)

An alternative method is *Modular Function Deployment* (MFD) (Ericsson and Erixon 1999), which creates a similar *Product Property Matrix* to identify platform modules (see Chaps. 4 and 24 for details and examples of MFD). Chapters 14 and 15 discuss QFD approaches to support platform optimization.

If existing products are available for analysis, an alternative approach for creating this matrix is to decompose the products into subsystems and components and then use a DSM, or *Design Structure Matrix* (Steward 1981; Eppinger et al. 1994; Browning 2001) to help identify modules. A component-based DSM for the UGV family is presented next in reference to the defining of a generic system architecture.

### 13.7 Generic System Platform Architecture Definition

A platform forms the base for a set of product variants; thus, the platform architecture should encompass all the variant architectures that intend to support. A simple way to create such a “generic” architecture for the platform is to create an architecture for each of the variants and merge them together. The variant architectures can be created using functional modeling or a component-based approach.

The functional modeling approach has benefit of being intuitive and visual, particularly for more complex block diagrams. In the functional approach (Pahl and Beitz 1996; Otto and Wood 2001), the recommended approach is to build each individual functional model and then merge these models into a generic platform architectural model. In the UGV example, the functional model in Fig. 13.5 is already a generic functional model since the individual UGVs differ in performance levels, not functionality, in the vertically leveraged scalable platform. For example, while each model may have a different size manipulator or camera, all of the UGVs have these components to perform common tasks of reaching/grasping objects and observing their surroundings, i.e., the functions are the same.

Rather than the functional block diagram approach, another possibility is to work with matrix methods and functions to extend the Generational Variety Index and Modular Function Deployment methods introduced in the previous section. Both methods can be used to map functions to components and thereby define modules, without explicitly defining a generic platform architecture. Module definition is discussed in more detail in the next section.

Finally, one can forgo the functional approach altogether and work directly with components, using the component-based Design Structure Matrix (DSM) approach. This avoids difficulties of thinking functionally but thereby also limits the space of design solutions to the components at hand. For demonstration, this component-based DSM approach is used for the UGV example (see Fig. 13.7). For the UGV example, the DSM is constructed using a teardown disassembly approach to system decomposition (Chiriac et al. 2011). In this case, four UGVs were disassembled;

**Fig. 13.7** Unclustered DSM for the UGV platform

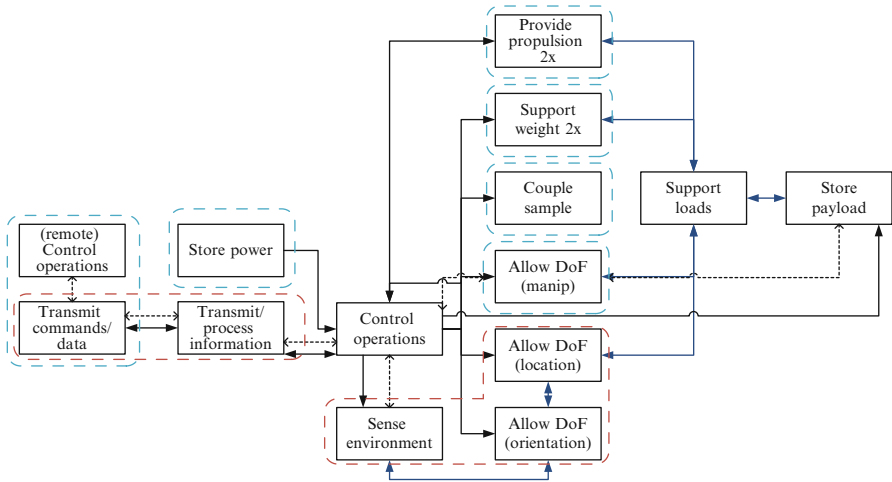
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1 Chassis			x	x			x	x				x		
2 Battery														
3 Flipper	x					x								
4 Main track	x					x								
5 Communication box						x							x	
6 Electronics box		x	x	x	x		x	x	x	x	x	x		
7 Manipulator	x					x								
8 Mast	x					x			x					
9 Head						x	x				x			
10 Gripper						x								
11 Cameras						x		x						
12 Payload Bay	x					x								
13 Antenna					x									x
14 OCU												x		

their subsystems or components were identified and then recorded to create the component-based DSM shown in Fig. 13.7. This DSM represents the “generic” platform architecture for the UGV family, which can now be used for module definition as discussed next.

### 13.8 Module Boundary Definition

There are many methods to define modules (Gershenson et al. 2004; Simpson 2004; Simpson et al. 2006). Some common methods include modularity heuristics (Stone et al. 2000; Zamirovski and Otto 1999), heuristic identification of commonalities using a modularity matrix (Dahmus et al. 2001), Modular Function Deployment (Ericsson and Erixon 1999) to identify modules by importance of components, and finally various numerical heuristic methods that operate on a DSM and cluster its elements into modules (Helmer et al. 2010; Yu et al. 2005; Yu et al. 2007). Most clustering algorithms include a measure to decide when to conclude the clustering, i.e., when the desired degree of modularity is achieved. Thebeau (2001) developed a measure to minimize connections outside modules whereas Yu et al. (2005) use an information theoretic approach to minimize the information needed to describe the connectivity between modules. Excellent reviews of both coupling and similarity modularity can be found elsewhere (Gershenson et al. 2004; Guo and Gershenson 2003). Hierarchical clustering can also be used to define modules, either based on the product requirements, component specifications (Hölttä-Otto et al. 2008), or in conjunction with another method that defines a metric, such as Modular Function Deployment (see Chaps. 4 and 24).

For the UGV example, we continue with both the functional model and the DSM to demonstrate alternative approaches to module boundary definition. First, using the functional model as a starting point, we use the module heuristics (Stone et al. 2000) to identify potential modules in the system architecture. As seen in Fig. 13.8,



**Fig. 13.8** Module definition using module heuristics

we identify multiple potential modules using the branching flow heuristic (blue dashed lines) as well as a transmission type module and a dominant flow module (both in red). The function structure is at a relatively high abstraction level, and thus the modules, for most part, include only one abstracted function. The modules defined in this approach are sensible and occur in current UGVs on the market. We note that if the system were to be decomposed to a further level of granularity, we would identify additional module candidates—such as all the separate drives (within the “Allow DoF” function) as conversion-transmission modules. We refer the reader to Chap. 9 for more discussion on the level of granularity and its impact on module identification.

Another approach to define module boundaries is to cluster the component DSM developed in Fig. 13.7. There are many clustering algorithms available (see <http://www.dsmweb.org>), but manual clustering is sufficient for the UGV example. Figure 13.9 shows the clustered DSM for the original UGV architecture in Fig. 13.7. This partitioning results in two bus modules (for the chassis and electronics), two  $3 \times 3$  modules (for communications and observation), and multiple modules that consist of a single component (for specific functionality, e.g., manipulator). Again, if the system was decomposed further, a larger DSM with larger modules would be created to identify modules within modules. For this system, one can compare Figs. 13.8 and 13.9 and see there is essentially the same result obtained. Both methods resulted in compounded components into a module, and both methods identified the chassis and electronics as buses (isolated modules with many parallel connections).

**Fig. 13.9** Clustered DSM for the UGV family

	1	6	2	13	5	14	3	4	7	8	9	11	10	12
1 Chassis	x						x	x	x	x				x
6 Electronics box		x		x			x	x	x	x	x	x	x	x
2 Battery			x											
13 Antenna					x	x								
5 Communication box		x		x										
14 OCU				x		x								
3 Flipper	x	x						x						
4 Main track	x	x							x					
7 Manipulator	x	x								x				
8 Mast	x	x									x			
9 Head		x								x		x		
11 Cameras		x									x		x	
10 Gripper		x												x
12 Payload Bay	x	x												

### 13.9 Architecture Roadmap

A platform should also support multiple product generations over time. A good way to design for future product variants is to plan for technology evolution. This can be done in several ways. In some methods such as MFD (Ericsson and Erixon 1999), this is taken into the account during module boundary definition. For example, in MFD a modularity rule states that if a component or subassembly is expected to evolve over time, it should be separated into a module. This concept generalizes into the idea of *technology roadmapping* (Albright 2002; Suh et al. 2010). In general, a *technology roadmap* is a plan to develop products in the future using technology that is not yet fully developed. Despite the incompleteness, the past trends allow for future prediction of expected performance. A common technology forecast is Moore’s law (Allan et al. 2002; Edenfeld et al. 2004) or similar performance improvement curves in, for example, telecommunication (Willyard and McClees 1987), printers (Ulrich and Eppinger 2004), and power transmission (Daima and Oliverb 2008).

In product platform design, such roadmaps are done at the module level, where each module is scrutinized for future evolution and strategically roadmapped. The system architecture of such modules now and into the future remains fixed, until the modules change so radically an entirely new platform is required (Suh et al. 2010). The module update cycle is usually faster than the platform redesign cycle.

For example for our UGV case, we consider the generic product architecture and the current modules sizes, and we can create a plan for evolving (or not evolving) the modules over time based on this. For example, we expect mobility technology (i.e., provide propulsion, support weight modules) to remain relatively constant in the near future, and thus there is no plan to evolve this module on a roadmap. The battery technology (i.e., store power module) and computing platform and embedded controls software (e.g., controls module), on the other hand, are expected to improve over time, and each UGV will need to be redesigned and upgraded as the

technology evolves. For example, battery energy per unit cost or per unit weight continually improves, and one can project increased stored energy expectations every 2 years. Comparing this against stored energy levels needed to achieve increased power loads of improving other modules (e.g., larger power train motors, increased imaging cameras, longer range communication) defines the point in time where it is useful to upgrade the battery module to higher power. Having such a roadmap will prevent the platform for hindering evolution and improvements as technology changes and new technologies are developed.

### 13.10 Commonality Assignment

The next step is to decide how many sizes of each module are needed, given the number of product variants and the modularity boundaries identified for the components. The maximum number of instances for any module is the number of product variants if every instance needs to be unique to every product variant (e.g., the chassis may be unique to the small, medium, and large UGVs). Conversely, the minimum number of instances for any module is one, i.e., one module that is common on all of the variants (e.g., the same camera on every UGV). Reality usually lies in between those two extremes where module instances may vary slightly across different subsets of products (e.g., the manipulator on the medium UGV may have two articulating segments while the large UGV may have three).

The Generational Variety Index (see Sect. 13.6) provides a starting point for commonality assignment. Low GVI values will not require much redesign within the product family; therefore, a single module may suffice for this component. Meanwhile, high GVI values require significant redesign indicating that multiple modules will be needed to achieve the range of requirements for the family. For the UGV example, a subsequent analysis of each pair of UGVs (e.g., small and medium, medium and large, and large and small) was conducted to translate the GVI recommendations in Fig. 13.6 to parametric variation needed in each subsystem. Through this analysis we sought to identify potential opportunities for scaling, for example, the chassis in one or more dimensions based on the threshold and objective values for each UGV pair even though the chassis will vary across each weight class. The final recommendations for commonality in key subsystems are listed in Table 13.2 where a “c” indicates where common settings may be used across two or more UGVs, e.g., chassis height can be common to all three UGVs, but only the small and medium have common chassis length and width based on the threshold and objective requirements.

In the approach outlined here, the determination of how many module sizes are needed is separated from the task of actually sizing each module. In Table 13.2, different sizes are shown for each of the product variants (small, medium, large), though the actual sizes of the modules is not yet determined. All that has been defined is the commonality. Further, for each architecture alternative, the allocation of Table 13.2 will change. Choice of module boundaries affects the choices of



**Table 13.2** Recommendations for subsystem commonality (Simpson et al. 2012)

Subsystem	Design parameters	Small	Medium	Large
Chassis	Length	c	c	
	Width	c	c	
	Height	c	c	c
Mobility	Wheels/tracks	c	c	c
	Wheel/track diameter			
	Wheel/track width			
Batteries	Length	c	c	c
	Width	c	c	c
	Mass	c	c	c
Manipulators	Outer arm radius	c	c	
	Arm segment length	c	c	
	Number of links	c	c	

number of module sizes. Mathematical models of the UGV performance could be combined with optimization algorithms to determine the best settings for each module and each parameter in the family, using these recommendations as a starting point. This is discussed in the next section; meanwhile, in Chap. 18 Khire et al. discuss how optimization can be used for commonality selection in more detail, and Simpson et al. (2012) provide additional details on the UGV example.

### 13.11 Architectural Module Sizing and Down Selection

At this point, there is a small set of alternative architecture platform concepts, each complete with a modularity scheme to implement the set of product variants. However, the modules have not been sized nor assessed for performance capability to correctly operate in each supported product variant. Since the module sizes are not yet known, the performance of each product variant as instantiated in each platform concept is not known. These must be computed or estimated before a down selection can be made amongst the platform concepts.

To do this, we now create equations of the requirements in terms of module sizing variables. For example, we can describe batteries with energy storage capacity, size, weight, etc. Using such variables, we can derive UGV system level equations of the UGV top speed, overall mass, climbing angle, etc. We term the module variables with  $x$ , UGV system responses with  $y$ , and the equations with  $f$ .

Given targets on each system responses  $y$  for each product variant, the shared modules can be sized for best sizes  $x$ . This is repeated for each platform architecture alternative, thereby quantifying their performance (on all variants) and allowing a well determined platform down selection.

**Table 13.3** Module sizes for a UGV family (Simpson et al. 2012)

Subsystem	Design parameters	Small	Medium	Large
Chassis	Length	0.56	0.59	0.66
	Width	0.23	0.22	0.30
	Height	0.32	0.33	0.34
Mobility	Wheels/tracks	Tracks	Tracks	Tracks
	Wheel/track diameter	0.26	0.26	0.26
	Wheel/track width	0.03	0.03	0.13
Batteries	Length	0.11	0.11	0.11
	Width	0.06	0.06	0.06
	Mass	1.40	1.40	1.40
Manipulators	Outer arm radius	0.02	0.02	0.02
	Arm segment length	0.57	0.52	0.31
	Number of links	3	3	3

For the UGV example, Table 13.3 shows the module sizes for one instance of the platform to achieve at least 80 % effectiveness across all of the requirements in each market segment. Table 13.3 is different for each alternative platform architecture, and a concept selection matrix can be used to rank each. The actual sizing of modules is, however, generally a nontrivial analysis. Typically, trade-off analyses are done and visualized for the non-dominated combinations of module variables that are for the Pareto frontier. Amongst this set, combinations can be sorted using judgment or a value-based gauge of the performance criteria. Methods to accomplish this are discussed by Khire et al. in Chap. 18, and more details on the trade-offs in the UGV family are discussed by Simpson et al. (2012).

## 13.12 Summary

Methods and tools to support product platform design have advanced remarkably over the past decade, and there is no single path to architecting a product platform. Some methods are more suitable for an approximate approach to get fast results, while others are better suited for more sophisticated analysis; however, most fall somewhere in between these two extremes.

Looking back at Fig. 13.1, we can see how one of the shortest paths to a platform is to build a DSM directly from components of existing products or from customer requirements using quality function deployment, for example, and then simply clustering the DSM to form modules. This is most likely suitable for a redesign of an existing family for more efficient variety and commonality ratio. For a more fundamental overhaul of a product platform, or a clean sheet design of a new product platform, other combinations of methods are likely more productive. Use of function-based methods helps abstract the problem and can lead to better solutions than simple reconfiguration of components.

Overall, we do not recommend or promote any one particular method over another nor even espouse the sequence through the methods as shown. Rather our goal has been to provide a reference for linking all of the different modular product platform development methods out there and in this book, along with where to find more information on these methods.

## References

- Albright RE (2002) Roadmapping for global platform products: product development and management association. *Visions Magazine* 26(4):19–22
- Allan A, Edenfeld D, Joyner WJ, Kahng A, Rodgers M, Zorian Y (2002) 2001 technology roadmap for semiconductors. *Computer* 35(1):42–53
- Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Trans Eng Manag* 48(3):292–306
- Chiriac N, Hölttä-Otto K, Suh E, Lysy D (2011) Three approaches to complex system decomposition. In: 13th international dependency and structure modelling conference, DSM'11, Cambridge, MA
- Churchill GA, Iacobucci D (2004) *Marketing research: methodological foundations*, 9th edn. South-Western College Publishing, ISBN: 0324201605
- Dahmus JB, Gonzalez-Zugasti JP, Otto KN (2001) Modular product architecture. *Des Stud* 22(5):409–424
- Daima TU, Oliverb T (2008) Implementing technology roadmap process in the energy services sector: a case study of a government agency. *Technol Forecast Soc Change* 75(5):687–720
- Edenfeld D, Kahng A, Rodgers M, Zorian Y (2004) 2003 technology roadmap for semiconductors. *Computer* 37(1):47–56
- Eppinger S, Whitney D, Smith R, Gebala D (1994) A model-based method for organizing tasks in product development. *Res Eng Des* 6(1):1–13
- Ericsson A, Erixon G (1999) *Controlling design variants: modular product platforms*. ASME Press, New York
- Gershenson JK, Prasad JK, Zhang Y (2004) Product modularity: measures and design methods. *J Eng Des* 15(1):33–51
- Guo F, Gershenson JK (2003) Comparison of modular measurement methods based on consistency analysis and sensitivity analysis. In: ASME 2003 design engineering technical conferences, Chicago, IL. ASME, Paper No. DETC2003/DTM-48634
- Griffin A, Hauser JR (1993) The voice of the customer. *Market Sci* 12(1):1–27
- Hauser JR, Clausing D (1988) The house of quality. *Harv Bus Rev* 66(3):63–73
- Helmer R, Yassine A, Meier C (2010) Systematic module and interface definition using component design structure matrix. *J Eng Des* 21(6):647–675
- Hölttä-Otto K, Tang V, Otto K (2008) Analyzing module commonality for platform design using dendrograms. *Res Eng Des* 19(2):127–141
- INCOSE (2010) *INCOSE systems engineering handbook v3.2*. International Council on Systems Engineering. <http://www.incose.org/>
- Kotler P, Keller KL (2009) *Marketing management*. Prentice Hall, Upper Saddle River
- Martin M, Ishii K (2000) Design for variety: developing standardized and modularized product platform architecture. *Res Eng Des* 13(4):213–235
- Meyer MH, Lehnerd AP (2000) *The power of product platforms*. The Free Press, New York, NY
- Moon SK, Kumara SR, Simpson TW (2006) Data mining and fuzzy clustering to support product family design. In: ASME design engineering technical conferences – design automation conference, Philadelphia, PA. ASME, Paper No. DETC2006/DAC-99287

- Otto K, Wood K (2001) Product design: techniques in reverse engineering, systematic design, and new product development. Prentice-Hall, New York, NY
- Pahl G, Beitz W (1996) Engineering design: a systematic approach. Springer, New York, NY
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf* 10(1):3–20
- Simpson TW, Siddique Z, Jiao J (2005) Product platform and product family design: methods and applications. Springer, New York, NY
- Simpson TW, Marion T, de Weck O, Holttä-Otto K, Shooter SB (2006) Platform-based design and development: current trends and needs in industry. In: ASME 2006 international design engineering technical conferences, Philadelphia, PA. Paper No. DETC2006/DAC-99229
- Simpson TW, Brennan S, Slingerland LA, Bobuk A, Logan D, Reichard K (2012) From user requirements to commonality specifications: an integrated approach to product family design. *Res Eng Des* 23(2):141–153
- Steward DT (1981) The design structure system: a method for managing the design of complex systems. *IEEE Trans Eng Manag* 28(3):71–74
- Stone RB, Wood KL, Crawford RH (2000) A heuristic method for identifying modules in product architectures. *Des Stud* 21(1):5–31
- Suh ES, Furst MR, Mihalyov KJ, de Weck O (2010) Technology infusion for complex systems: a framework and case Study. *Syst Eng* 13(2):186–203
- Thebeau RE (2001) Knowledge management of system interfaces and interactions for product development process. System design & management program. M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA
- Ulrich KT, Eppinger SD (2004) Product design and development, 3rd edn. McGraw-Hill, New York, NY
- Willyard CH, McClees CW (1987) Motorola's technology roadmap process. *Res Manag* 30(5):13–19
- Yu TL, Yassine A, Goldberg DE (2005) An information theoretic method for developing modular architectures using genetic algorithms. University of Illinois, Department of General Engineering. Urbana-Champaign: Illinois Genetic Algorithms Laboratory IlliGAL
- Yu TL, Yassine AA, Goldberg DE (2007) An information theoretic method for developing modular architectures using genetic algorithms. *Res Eng Des* 18(2):91–109
- Zamirowski EJ, Otto KN (1999) Identifying product family architecture modularity using function and variety heuristics. In: ASME design engineering technical conferences – 11th international conference on design theory and methodology, Las Vegas, NV. ASME, Paper No. DETC99/DTM-8760
- Zhang Y, Jiao J, Ma Y (2007) Market segmentation for product family positioning based on fuzzy clustering. *J Eng Des* 18(3):227–241

# Chapter 14

## A QFD-Based Optimization Method for Scalable Product Platform

Xinggong Luo, Jiafu Tang, and C.K. Kwong

**Abstract** In order to incorporate customer into the early phase of the product development cycle and to better satisfy customers' requirements, this research adopts quality function deployment (QFD) for optimal design of scalable product platform and product family. A five-step QFD-based method is proposed to determine the optimal values for platform engineering characteristics (ECs) and non-platform ECs of the products within a product family. First of all, the houses of quality (HoQs) for all product profiles are developed and a QFD-based optimization approach is used to determine the optimal ECs for each product profile. Sensitivity analysis is performed for each EC with respect to overall customer satisfaction (OCS). Based on the obtained sensitivity indices of ECs, a mathematical model is established to simultaneously optimize the values of the platform and the non-platform ECs. Finally, by comparing and analyzing the optimal solutions with different number of platform ECs, the ECs with which the worst OCS loss can be avoided are selected as platform ECs. An illustrative example is used to demonstrate the feasibility of this method. A comparison between the proposed method and a two-step approach is conducted on the example.

---

X. Luo (✉) • J. Tang

State Key Lab of Synthetic Automation of Process Industries, School of Information Science and Engineering, Northeastern University, Shenyang, Liaoning 110004, People's Republic of China  
e-mail: [xgluo@mail.neu.edu.cn](mailto:xgluo@mail.neu.edu.cn)

C.K. Kwong

Department of Industrial and System Engineering, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

## 14.1 Introduction

Nowadays, the growing demand for diversified and personalized products has imposed a necessity on manufacturing companies to provide a large variety of products in order to guarantee customer satisfaction. To achieve this, many companies are using platform-based product development to realize families of products with sufficient variety to meet customers' demands while keeping costs relatively low (Thevenot and Simpson 2006). Actually, product platform technology offers a multitude of benefits, such as increased flexibility and responsiveness, reduced development time and system complexity, reduced development and production costs, and improved ability to upgrade products (Simpson et al. 2006).

Developing a product family is a creative activity that involves designers drawing upon numerous types of information, including both product–customer requirements and design developments in related fields (Lin et al. 2008). In recent years, the necessity to better satisfy customer's need is emphasized by more and more industries, resulting in prevailing of the concept of market-pull strategies for product development (Tang et al. 2002). In current highly competitive and dynamic marketplaces, only effective customer-driven strategies on product development are able to assist a company to gain a prominent competitive edge over its competitors; therefore, designers are responsible for generating customer-driven products to maximize customer satisfaction (Khoo et al. 2002).

A widely used customer-driven methodology for new product development is quality function deployment (QFD) (Hauser and Clausing 1988). QFD helps the product development team systematically relate the customer attributes that represent the overall customer concerns to the design requirements and thus aims at the satisfaction of the customer's need at the very beginning of product design (Karsak et al. 2003).

A number of methodologies for optimal design of product family based on scalable product platform have been proposed in published research papers. A brief review of these papers is given in Sect. 14.2. One can see that the objectives of these methodologies are to maximize performance of product family, balance commonality and distinctiveness of product platform, minimize total engineering cost, and maximize profit or revenue from the market. In this research, we aim at maximizing the overall customer satisfaction (OCS) towards a product family by applying QFD methodology. The intent of applying QFD is to incorporate customer into the early phase of the product development cycle for product family through marketing surveys and interviews and to assume the achievement of customer-required quality (Wang 1999). The main contribution of this research is that it introduces the product quality criterion OCS into the optimization process of product platform and product family design by integrating QFD method.

The rest of this chapter is organized as follows. Section 14.2 reviews the relevant literature on product platform optimization and QFD. Section 14.3 specifies the main procedures and the key steps of the QFD-based optimization method.

Section 14.4 illustrates a case study to empirically verify the feasibility and effectiveness of the method. The characteristics of this method, limitations, and some future research potentials are discussed in Sect. 14.5.

## 14.2 Related Work

In this section, some literature highly related to this study is briefly reviewed under the two classes: optimization for scalable product platform and quality function deployment.

### 14.2.1 *Optimization for Scalable Product Platform*

A product platform is defined as a set of common components, modules, or parts from which a stream of derivative products can be efficiently developed and launched (Meyer and Lehnerd 1997). Meanwhile, products that share a common product platform but have specific features and functionality to satisfy different sets of customers form a product family (Meyer and Utterback 1993). In general, there are two kinds of product families (Simpson et al. 2006): one is module-based (or configurable) product family wherein product family members are instantiated by adding, substituting, and/or removing one or more functional modules from the platform and another one is a scalable (or parametric) product family, in which several key engineering characteristics (ECs) are used to scale the product platform to form a variety of products.

A large quantity of research papers have been published on design of product family and platform, emphasizing on different perspectives such as customer engineering, product family positioning, platform optimization, business strategy, manufacturing and production, information technology, and general management. Discussion and classification of these research papers can be found in the recent survey papers by Jiao et al. (2007), Simpson (2004), and Fujita (2002). In this research, we only concern the optimization of scalable product platform; hence we narrow the review of this section accordingly.

In design of scalable platform, all members share a parametric description, i.e., they are described by the same ECs, but the ECs may take on different instantiated values for different product profiles (a product profile is a member of a specified product family). The objective of the optimization is to help designers to determine the best EC settings for a product platform and the product profiles. Generally, the design and optimization of scalable product platform include the following main stages (Dai and Scott 2007):

1. Platform configuration, i.e., how to identify which ECs should be taken as platform ECs.

2. Determine the appropriate values for the selected platform ECs.
3. Determine the optimal values for the non-platform ECs of the product profiles,

where 2 and 3 are combined into one in some approaches.

One of the first systematic design methods for scalable product platforms is the Product Platform Concept Exploration Method (PPCEM) proposed by Simpson et al. (1999). In PPCEM, a compromise Decision Support Problem (DSP) is used to model the necessary constraints and goals for product platform. The model is a multi-objective one and eventually a set of conflicting goals with minimal loss in performance is achieved. Conner et al. (1999) develop the Variety Tradeoff Evaluation Method (PVTEM) for product platform optimization. Similar to PPCEM, PVTEM assesses the appropriate product family trade-offs using the commonality and performance indices.

Moore et al. (1999) apply conjoint analysis to help designing product platforms by bringing together demand-side forecasting methods with supply-side cost estimates. Their empirical research further shows the importance of consideration of commonality of products. However, the details of the optimization process and models are not provided in the paper. Based on PPCEM, Nayak et al. (2002) present the Variation-Based Platform Design Method (VBPDM), which uses variation-based modeling to accommodate flexible design specifications to integrate the selection of common or platform parameters and the scale factors as part of the commonality and performance trade-off. Messac et al. (2002) propose using physical programming and PPCEM to design product families. The physical programming method formulates the optimization problem in terms of physically meaningful terms and parameters and hence facilitates trade-off analysis and decision making during product family design. Fellini et al. (2004) suggest a two-stage methodology for making commonality decisions based on the individual optima and sensitivity analysis of functional requirements. Their methodology uses the first-order information obtained from individual design optimizations to compute a metric for performance deviations. Different from the optimization approaches targeted at product performance goals, Kumar et al. (2006) propose the Market-Driven Product Family Design (MPED) methodology to model the optimization of product platform with considering product-line positioning. Nested logit choice rule is used in their model to simulate consumer behavior in a multi-segment market with competitive products. Dai and Scott (2007) propose a six-step method which incorporates sensitivity analysis and cluster analysis to solve the platform configuration problem and to achieve the value settings for platform and non-platform ECs. Williams et al. (2007) develop a Product Platform Constructal Theory Method (PPCTM) to help designers to systematically manage modularity and commonality in development of platforms while handling issues of multiple levels of commonality, multiple product specifications, and the inherent trade-offs between platform extent and performance. Their PPCTM is able to handle multiple design objectives and nonuniform demand in a market by infusing the utility-based compromise Decision Support Problem and demand modeling techniques.



However, all the abovementioned methods for scalable product platform do not consider the metric of product quality in the design optimization process. This research will fill this gap by integrating QFD into the design optimization of scalable product platform.

### 14.2.2 *Quality Function Deployment*

The basic concept of QFD is to utilize a set of charts called the houses of quality (HoQ) (Hauser and Clausing 1988) to translate customer requirements (CRs) into ECs and subsequently into parts, characteristics, process plans, and manufacture operations. A house of quality typically contains information on “what to do” (*whats*), “how to do it” (*hows*), the integration of this information (relationships between CRs and ECs and among ECs), and benchmarking data (Kim et al. 2000). Based upon the information contained in a HoQ, the target values for the ECs of a product can be determined to achieve a high level of OCS. Because we propose a QFD-based approach to achieve the optimal values for platform and non-platform ECs, here we only briefly review the QFD literature related to the following two aspects (1) setting target ECs values and (2) product family and platform.

Regarding setting target EC values, although the amount of literature on QFD is vast, there are only a few research papers focusing on systematic procedures for setting the target EC values.

The first prescriptive modeling approach is given by Wassermann (1993), who formulates the QFD planning process as a linear programming model to select the mix of design features with the highest level of customer satisfaction. Moskowitz and Kim (1997) propose a QFD-based decision support prototype for optimizing product designs based upon an integrated mathematical programming approach. Fung et al. (1998) suggest a fuzzy QFD model to facilitate the design decision on target values for ECs with the use of a fuzzy rule base. Park and Kim (1998) present an integrated decision model for selecting the optimal ECs by using a modified HoQ. Kim et al. (2000) propose a fuzzy multi-criteria modeling approach for QFD planning using fuzzy linear models with symmetric triangular fuzzy number coefficients. Tang et al. (2002) and Fung et al. (2002) consider a fuzzy formulation combined with a genetic-based interactive approach to QFD planning and develop fuzzy optimization models to determine target values of ECs with financial consideration. Chen et al. (2005) utilize a fuzzy expected value operator to model QFD planning in a fuzzy environment and further propose an approach for determining the target values of ECs with the help of two fuzzy expected value models.

On the other hand, most of the research papers related to QFD focus on the design and manufacturing processes of a particular product; only a few of them concern QFD in design of product family and platform.

Among these exceptions, Cohen (1995) suggests a planning matrix used for strategic goal setting, through which the impact of the product family members on

meeting needs determines the priorities of the product family members. Erixon et al. (1996) propose using QFD to clarify the customer requirements and modularize a product. Martin and Ishii (2002) apply a modified QFD structure to generate a Generational Variety Index (GVI) for a product platform. Their approach is a two-phase QFD translation (1) translating the subjective customer requirements into quantifiable engineering specifications and (2) translating the engineering metrics to the components used in the design. Hsiao and Liu (2005) introduce a modified QFD in the three-stage product family design methodology, in which the matrix only lists the differences in customer requirements rather than common requirements, in order to identify crucial and meaningful design changes from the perspective of customers. Kim et al. (2006) apply a QFD-based multi-attribute optimization model for calculating the sensitivity indices when determining the product platform elements. Jariri and Zegordi (2007) establish a QFD-based mathematical programming model to identify the proper subsystem for each platform system. These systems can be chosen from several commercially available alternatives, and the decision variables of their model are the percents of alternatives in each system.

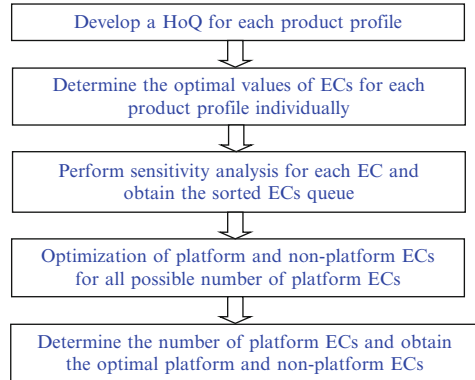
### ***14.2.3 Commonality Indices***

A commonality index is a metric to measure the degree of commonality within a product family based on different parameters such as the number of common components, the component costs, and the manufacturing processes (Thevenot and Simpson 2006). The larger a commonality index is, the more components and manufacturing processes a product family shares and the more cost is possible to be saved for a company. Introduction of the six commonly used commonality indices (Degree of Commonality Index, Total Constant Commonality Index, The Product Line Commonality Index, Percent Commonality Index, Commonality Index, Component Part Commonality Index) and their comparison can be found in Thevenot and Simpson's research paper (Thevenot and Simpson 2006).

## **14.3 A Method for Product Platform Optimization Based on QFD**

In this section, a five-step QFD-based method is proposed to determine the rational number of platform ECs and the optimal values of platform and non-platform ECs. The procedure of the proposed method is explained in Sect. 14.3.1. Some key steps of the method including modeling of the optimization problem and sensitivity analysis are explained in Sects. 14.3.2–14.3.4.

**Fig. 14.1** Flow of the proposed method



### 14.3.1 Procedure

The flow of the QFD-based method for product platform optimization is shown in Fig. 14.1. The five steps of the proposed method are explained as follows:

- Step 1: Develop a HoQ for each product profile. This mainly involves identifying and prioritizing CRs; generating ECs; determining relationships between CRs and ECs, and correlations among ECs; and conducting competition benchmarking of competitive products (Madu 2006). The HoQs of the products in a family have the same house structure, house roof (ECs correlation matrix), and binary relationships between CRs and ECs, but their benchmarking scores of the existing products (representing the customer perception of the competitors' product and the firm's existing products on CRs) and the CRs–ECs relationship matrix (in which an element represents the quantitative level of strength of the relationship between a CR and a EC) may be different from each other.
- Step 2: Determine the optimal values of ECs for each product profile. A QFD-based optimization method can be applied to achieve the optimal values of ECs of a product profile. In light of the HoQs established in Step 1, the optimization model described in Sect. 14.3.2 is solved individually for each product in the family.
- Step 3: Perform sensitivity analysis for each EC. Sensitivity analysis described in Sect. 14.3.3 is carried out to calculate the global sensitivity index (SI) of an EC. After the global SIs of all ECs are computed, sort the ECs into a queue according to their global SIs in ascending order. The ECs in forefront of the queue are inclined to be platform ECs because relatively little customer satisfaction is lost as a result of commonization. Leave the number of platform ECs as a variable for determination in the subsequent steps. Put the first element of the queue into  $U$ .
- Step 4: Optimization of platform and non-platform ECs. By applying the QFD-based optimization model described in Sect. 14.3.4, for a given set of

platform ECs  $U$ , the optimal values for product platform ECs and other non-platform ECs for product profiles are obtained at the same time.

Step 5: Put the next element into  $U$  and perform Step 4; repeat this process until all ECs are appended into  $U$ . Compare and analyze the optimal solutions with different number of platform ECs, and determine the number of platform ECs with which the worst OCS loss can be avoided. In Sect. 14.4, we show an example of determination of the number of platform ECs.

### 14.3.2 Determine the Optimal Values of ECs for Each Product Profile

Optimization approaches based on QFD (Fung et al. 1998; Kim et al. 2000; Tang et al. 2002) can be applied to determine the optimal ECs for each product profile. Assume that in this step, all the work related to construction of the HoQ has been accomplished. Suppose that in a product,  $m$ CRs denoted by  $cr_i$  ( $i = 1, 2, \dots, m$ ) and  $n$ ECs denoted by  $ec_j$  ( $j = 1, 2, \dots, n$ ) are being considered. The  $m$ CRs can be determined by a market survey conducted by the producing company. The available methods include focus group, individual interviews, listening and watching, and using existing information. On the other hand, the  $n$ ECs are selected by the company's technicians or product development team. Usually the ECs can be generated from the current product standards or selected by ensuring through cause-effect analysis that the ECs are the first-order causes for the CRs (Akao 1990).

Let  $x_j$  be the normalized target value of  $ec_j$  and  $y_i$  be the customer perception of the degree of achievement of  $cr_i$ ; the functional relationship between  $y_i$  and ECs can be expressed as

$$y_i = f_i(x_1, x_2, \dots, x_n) \quad (14.1)$$

On the other hand, there are correlations among the ECs. The functional relationship between  $x_j$  and other ECs can be expressed as

$$x_j = g_j(x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n) \quad (14.2)$$

Let  $S$  be the degree of OCS to the product; the process of determining  $x_j$  ( $j = 1, 2, \dots, n$ ) in QFD can be formulated as an optimization problem as follows:

$$\max S = h(y_1, y_2, \dots, y_m) \quad (14.3)$$

subject to

$$y_i = f_i(x_1, x_2, \dots, x_n), (i = 1, 2, \dots, m) \quad (14.4)$$

$$x_j = g_j(x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_n), (j = 1, 2, \dots, n) \quad (14.5)$$

where  $h$  is a function for mapping  $y_i$  ( $i = 1, 2, \dots, m$ ) to  $S$ . The determination of  $f_i$ ,  $g_j$ , and  $h$  are described in Sect. 14.3.4, where the joint optimization model uses the similar functions. Although these functions can be in complex forms, in this research we consider them as linear forms for simplicity.

By applying this optimization model, the optimal values of ECs for each product profile can be found individually.

### 14.3.3 Sensitivity Analysis for Determining Platform ECs

Based on the achieved optimal values of ECs, sensitivity analysis can be performed for each EC with respect to OCS. The metric of sensitivity, called sensitivity index (SI), is measured as the ratio of the change in OCS to the change in an EC. In light of the calculation formula provided by Dai and Scott (2007), the SI of  $ec_j$  for a selected product profile, called local SI, can be calculated as follows:

$$SI_{x_j} = \frac{(S^* - S_{x_j^* + \Delta x_j}^*) + (S^* - S_{x_j^* - \Delta x_j}^*)}{2\Delta x_j} \quad (14.6)$$

where  $x_j^*$  is the optimal value of  $x_j$  obtained in Sect. 14.3.2;  $\Delta x_j$  is a very small number so that a first-order Taylor expansion can be used to approximate the first derivative of a curve;  $S^*$  is the value of objective function of the optimal solution, i.e., the maximal degree of OCS for this selected product profile; and  $S_{x_j^* + \Delta x_j}^*$  and  $S_{x_j^* - \Delta x_j}^*$  are the new optimal  $S$  when  $x_j^*$  is changed to  $x_j^* + \Delta x_j$  and  $x_j^* - \Delta x_j$ , respectively. Note that other ECs, i.e.,  $x_{j'}$  ( $j' = 1, 2, \dots, j-1, j+1, \dots, n$ ), may have new values when  $x_j^*$  is changed.

After the local SIs of  $ec_j$  for all product profiles are calculated according to Eq. (14.6), the global SI of  $ec_j$  can be determined as the weighted average of these local SIs. Actually, the global SI of  $ec_j$  indicates how sensitive the OCS is to changes in  $x_j$ . In other words, a low global SI of  $ec_j$  means that relatively little customer satisfaction will be lost if  $x_j$  is adjusted to a common value for a group of product profiles. Therefore, all those ECs with low global SIs are regarded as possible platform ECs. Based on the obtained global SIs, the platform ECs can be worked out by using clustering analysis (Dai and Scott 2007) or heuristic methods. Clustering analysis may be hierarchical clustering or  $k$ -means clustering and a widely used tool providing these functions is software SPSS. In the process of clustering analysis, the ECs with the lower global SIs are grouped into a cluster as platform ECs, while the ECs with the higher global SIs are grouped into another cluster as non-platform ECs. Heuristic methods, on the other hand, aim at finding a

threshold value for platform ECs so that the ECs with global SIs lower than the threshold are recognized as platform ECs and the other ECs as non-platform ECs. Simple heuristic methods to determine the threshold value include averaging global SIs of all ECs or weighted-sum technique. It should be noted that, however, the determination of platform ECs is a trial-and-error step and human interaction is involved in the process (Fellini et al. 2004). In this research we propose a method with considering the number of platform ECs as a variable, which is described in Steps 3, 4, and 5 in Sect. 14.3.1.

For each determined platform EC, the EC values of the product profiles are usually grouped into one group or several groups; in each group, a shared value is used for this EC as a result of commonization. The groups of EC values can be determined by using hierarchical clustering analysis. The optimal values of an EC of the product profiles are distributed with different density along the range of EC values. In the process of clustering, those product profiles with closer EC values are grouped into a cluster so that less customer satisfaction loss occurs from the commonization of EC values. The total number of clusters is then determined by observing the agglomerative result of clustering analysis. Similar to the determination of platform ECs, this is also a trial-and-error step: if the results are unsatisfactory, a new set of clusters may be explored.

### 14.3.4 Optimization of Platform and Non-platform ECs

The next step is to set the appropriate values for platform ECs and to determine the optimal values of other non-platform ECs for each product profile. A QFD-based optimization model is established to set the optimal values for these platform and non-platform ECs at the same time.

#### 14.3.4.1 Problem Definition

Suppose that in a company, there are  $K$  product profiles denoted by  $pv_k$  ( $k = 1, 2, \dots, K$ ). Note that the number of product profiles ( $K$ ) is given a priori. A product profile is similar to other product profiles in functions and structures. Therefore, the same  $cr_i$  ( $i = 1, 2, \dots, m$ ) and  $ec_j$  ( $j = 1, 2, \dots, n$ ) are considered for all of the product profiles.

In order to establish a product platform for these product profiles, sensitivity analysis and clustering analysis are performed to select the platform ECs and to determine the groups of EC values for each platform EC in advance. After these analysis processes are completed, suppose that the set of the indices of the determined platform ECs is denoted by  $U$ ; for each  $ec_{j'}$  ( $j' \in U$ ), there are  $n_{j'}$  groups of target values of  $ec_{j'}$  (i.e., there are  $K$  target values of  $ec_{j'}$  corresponding to the  $K$  product profiles, and they are grouped into  $n_{j'}$  groups); and for the  $l$ th ( $l = 1, 2, \dots, n_{j'}$ )

group, the set of the indices of the corresponding product profiles is denoted by  $V_{j'l}(V_{j'l} \cup V_{j'2} \cup \dots \cup V_{j'n_{j'}} = \{1, 2, \dots, K\})$ . In the  $l$ th ( $l = 1, 2, \dots, n_{j'}$ ) group, a shared value of  $ec_{j'}$  denoted by  $\hat{x}_{j'l}$  is to be determined to replace all values of  $ec_{j'}$  in the  $l$ th group.

For instance, a family of products has five product profiles ( $K = 5$ ) and four ECs (EC1, EC2, EC3, and EC4). After clustering analysis, EC3 and EC4 are selected as platform ECs ( $U = \{3, 4\}$ ). The optimal values of EC3 of the five product profiles are very close and they are clustered into one group ( $n_3 = 1, V_{31} = \{1, 2, 3, 4, 5\}$ ). The optimal values of EC4 are clustered into two groups: the first two product profiles are in a group and the others are in the other group ( $n_4 = 2, V_{41} = \{1, 2\}$ , and  $V_{42} = \{3, 4, 5\}$ ). There are three shared ECs required to be determined ( $\hat{x}_{31}, \hat{x}_{41}$ , and  $\hat{x}_{42}$ ).

Let  $x_j^k$  be the normalized target value of  $ec_j$  of  $pv_k$ ,  $y_i^k$  be the customer perception of the degree of achievement of  $cr_i$  of  $pv_k$ ,  $f_i^k$  be the functional relationship between  $y_i^k$  and ECs of  $pv_k$ , and  $g_j^k$  be the functional relationship between  $ec_j$  and other ECs of  $pv_k$ . The goal of the optimization problem is to determine the target value of ECs of product profiles with the objective of maximizing the average degree of customer satisfaction.

The optimization model of the problem can be formulated as

$$\max S' = \sum_{k=1}^K h(y_1^k, y_2^k, \dots, y_m^k) / K \quad (14.7)$$

subject to

$$y_i^k = f_i^k(x_1^k, x_2^k, \dots, x_n^k), (i = 1, 2, \dots, m; k = 1, 2, \dots, K) \quad (14.8)$$

$$x_j^k = g_j^k(x_1^k, \dots, x_{j-1}^k, x_{j+1}^k, \dots, x_n^k), (j = 1, 2, \dots, n; k = 1, 2, \dots, K) \quad (14.9)$$

$$x_{j'}^{k'} = \hat{x}_{j'l}, (\forall k' \in V_{j'l}, l = 1, 2, \dots, n_{j'}, j' \in U) \quad (14.10)$$

$$\hat{x}_{j'l}^L \leq \hat{x}_{j'l} \leq \hat{x}_{j'l}^U, (\forall k' \in V_{j'l}, l = 1, 2, \dots, n_{j'}, j' \in U) \quad (14.11)$$

where  $\hat{x}_{j'l}^L$  and  $\hat{x}_{j'l}^U$  are the lower and upper bound of  $\hat{x}_{j'l}$ , respectively. Additional product platform constraints may be added to the above formulation as appropriate.

#### 14.3.4.2 Objective Function

The objective function of the model represents the average value of the degree of customer satisfaction to all product profiles. On the other hand, for a particular

product profile,  $pv_k, h(y_1^k, y_2^k, \dots, y_m^k)$  can be obtained by aggregating the degrees of customer satisfaction with individual CRs (Chen et al. 2004):

$$h(y_1^k, y_2^k, \dots, y_m^k) = \sum_{i=1}^m w_i^k s_i(y_i^k) \tag{14.12}$$

where  $w_i^k$  is the scaled relative importance of  $cr_i$  of  $pv_k$  ( $0 \leq w_i^k \leq 1, \sum_{i=1}^m w_i^k = 1$ ) and  $s_i$  is an individual value function on  $cr_i$ . For each CR of  $pv_k$  ( $k = 1, 2, \dots, K$ ),  $y_i^k$  ( $i = 1, 2, \dots, m$ ) can be assigned as a numerical value to indicate the degree of satisfaction of  $cr_i$  in comparison with the competitors. This numerical value can be chosen from a positive scale [a,b] (e.g., 1–5). Therefore,  $s_i(y_i^k)$  can be scaled in such a way that  $s_i(y_i^{\min}) = 0$  and  $s_i(y_i^{\max}) = 1$  and can be configured as

$$s_i(y_i^k) = (y_i^k - y_i^{\min}) / (y_i^{\max} - y_i^{\min}) \tag{14.13}$$

and hence  $h(y_1^k, y_2^k, \dots, y_m^k)$  can be rewritten as follows:

$$h(y_1^k, y_2^k, \dots, y_m^k) = \sum_{i=1}^m w_i^k (y_i^k - y_i^{\min}) / (y_i^{\max} - y_i^{\min}) \tag{14.14}$$

and thus  $h(y_1^k, y_2^k, \dots, y_m^k)$  is also a value between 0 and 1, with 0 being the worst and 1 the best. Therefore, the objective function of the model can be expressed as

$$S' = \sum_{k=1}^K \sum_{i=1}^m w_i^k (y_i^k - y_i^{\min}) / (y_i^{\max} - y_i^{\min}) / K \tag{14.15}$$

### 14.3.4.3 Constraints

In the aforementioned model, constraint (14.10) ensures that in any group for each platform EC, all values of ECs are the same and equal to a shared value. Constraint (14.11) confines the bound of the shared value in each group. Constraint (14.8) describes the functional relationship between  $y_i^k$  and ECs of  $pv_k$ . A commonly used method to formulate this functional relationship is to use regression-based methods to estimate the parameters of this functional relationship (Kim et al. 2000; Fung et al. 2002; Tang et al. 2002). Usually, for simplicity, this functional relationship can be considered as a linear function as follows:

$$y_i^k = a_{i,0} + a_{i,1}x_1^k + a_{i,2}x_2^k + \dots + a_{i,n}x_n^k \tag{14.16}$$



where  $A_i = (a_{i,0}, a_{i,1}, a_{i,2}, \dots, a_{i,n})$  is a vector of coefficients to be estimated. These coefficients can also be further defined as fuzzy parameters to represent the inherent fuzziness in QFD.

Because a HoQ provides effective information on the basic relationships between CRs and ECs, relationships among ECs, and the benchmarking set, such information can be used as the base for estimation. For example, from the matrices of HoQ, one can notice that some ECs may not have any relationship with a CR; this indicates that the coefficients corresponding to these ECs in Eq. (14.16) can be set to zero directly. Suppose that there are  $H$  competitors involved in HoQ and  $X_h^k = (x_{1,h}^k, x_{2,h}^k, \dots, x_{n,h}^k)^T$  ( $h = 1, 2, \dots, H$ ) is the real-valued input vector of the normalized target values of ECs of the  $h$ th competitor; the set of data points for the regression can be defined as

$$P = \{(X_h^k, y_{i,h}^k) | h = 1, 2, \dots, H\} \quad (14.17)$$

where  $y_{i,h}^k$  is the degree of customer satisfaction of the  $h$ th competitor on  $cr_i$ . By using these data points, the coefficients in Eq. (14.16) can be estimated via least-square regression technique.

Similarly, constraint (14.9) describing the functional relationship between  $ec_j$  and other ECs can also be formulated as a linear function by using the regression-based method.

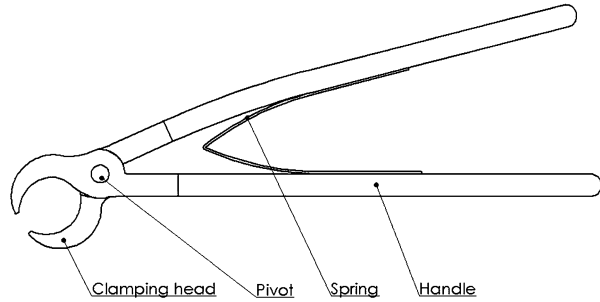
The optimization model established in this section is a standard linear programming one and can be solved by any linear programming software package. Due to the high efficiency of simplex algorithm, the computation time for solving the model is tolerable even for large-scale problem. However, if there are large quantities of product profiles, the work related to construct the optimization model (such as developing HoQs of product profiles and performing regression and sensitivity analysis) may be tedious and time-consuming. It would be a good idea to develop a software package to automatically establish HoQs, perform regression, and generate SIs for large-scale applications.

## 14.4 Case Study and Analysis

### 14.4.1 Case Description

An example of an industrial pincers, which is a product of an oil equipment corporation in Jiangsu province, China, is introduced in this section to illustrate the application of the proposed method. The pincers as shown in Fig. 14.2 is a scalable product, i.e., a similar product can be developed by scaling the values of one or more ECs of an existing product. These two products have similar functions and structures, but their performances vary to satisfy different requirements

**Fig. 14.2** Schematic diagram of an industrial pincers



of customers. The corporation has a series of pincers products with various sizes, including pincers A, B, C, D, and E. In order to reduce the cost of design and manufacturing, the corporation is planning to reduce variety in products and develop a product platform for the pincers.

According to the survey in the marketplace, four major CRs are identified to represent the biggest concern of the customers of the pincers. They are “strong clamping force” (CR1), “long durability” (CR2), “light weight” (CR3), and “low cost” (CR4). In light of the engineer’s design experiences of the product, four ECs are identified, i.e., “size of clamping head” (EC1), “length of handle” (EC2), “diameter of pivot” (EC3), and “thickness of antirust coat” (EC4). Four main competitors of the corporation, Co1, Co2, Co3, and Co4, are considered. The binary relationship between CRs and ECs, the correlation among ECs, and technical measure data collected from the corporation and the main competitors of pincers A are illustrated in HoQ in Fig. 14.3. The roof of the HoQ with black dots indicates that EC1 and EC2 have correlations with each other. The middle part of the HoQ with black dots shows the relationship between the CRs and ECs. The left part of the HoQ shows the weights of CRs. The lower part of the HoQ shows the ECs values of the products. The right part of the HoQ shows the benchmarking information of the products. Since most of the information in HoQ is the same for pincers product A, B, C, D, and E (e.g., the relationship between the CRs and ECs), the HoQs of pincers B, C, D, and E are not provided. However, the different information in HoQ, i.e., the engineering measures and benchmarking information of pincers B, C, D, and E, are provided in Table 14.1.

### 14.4.2 Computational Results

The coefficients of the relationship functions and correlation functions are obtained by using the least-square regression method provided in Microsoft Excel software package. The calculation results are listed in Table 14.2. Based on the parameters in HoQs and the coefficients, five QFD-based optimization models are established for pincers, A, B, C, D, and E, respectively. Software ILOG CPLEX is applied to solve

	ECs	EC <sub>1</sub>	EC <sub>2</sub>	EC <sub>3</sub>	EC <sub>4</sub>	Correlation	Benchmark information					
	EC <sub>1</sub>		•									
	EC <sub>2</sub>	•										
	EC <sub>3</sub>											
	EC <sub>4</sub>											
CRs	Weights	Relation				Ours	Co1	Co2	Co3	Co4	Min	Max
CR <sub>1</sub>	0.35	•	•			2.7	4.5	4.2	3.6	3.1	1	5
CR <sub>2</sub>	0.3	•		•	•	3.5	2.6	3.1	4.6	2.5	1	5
CR <sub>3</sub>	0.15	•	•			4.1	3.4	3.5	3.7	3.9	1	5
CR <sub>4</sub>	0.2	•	•		•	2.1	2.5	2.0	0.2	3.2	1	5
	Units	mm	mm	mm	μm	50.6	58.8	57.4	56.4	51.4	Satisf.(%)	
	Ours	185	860	24	40	Engineering Measures						
	Co1	198	890	28	20							
	Co2	195	885	27	30							
	Co3	192	875	26	60							
	Co4	188	865	25	20							
	Min	180	850	20	20							
	Max	220	950	40	60							

Fig. 14.3 HoQ of pincers A

these linear programming models. The obtained optimal values of the ECs are listed in Table 14.3.

Sensitivity analysis is performed at the optimal point for each EC. The local SIs and the global SIs of the ECs calculated according to Eq. (14.6) are given in Table 14.4.

Following the steps described in Sect. 14.3.1, we sort the ECs as {EC3, EC4, EC1, EC2} according to the global SIs in ascending order. By setting the number of platform ECs as 0, 1, 2, 3, and 4, respectively, the corresponding optimization models for maximizing the average degree of OCS are established. Again software ILOG CPLEX is applied to solve the linear programming models. Figure 14.4 shows the calculated maximal OCS and OCS loss with different number of platform ECs. One can see that the worst OCS loss (0.083) occurs when the number of platform ECs is shifted from two to three, which implies that setting the number of platform ECs as two is rational in order to avoid this OCS loss. Therefore, EC3 and EC4 are identified as platform ECs.

From Fig. 14.4 it can be observed that if EC3 and EC4 which have relatively low global SIs are selected as platform ECs, only 0.005 OCS is lost due to commonization. On the other hand, if EC1 and EC2 which have relatively high global SIs are set as platform ECs, 0.107 OCS is lost, which is much higher than that of EC3 and EC4. This result empirically shows the feasibility of the sensitivity analysis in the optimization process.

**Table 14.1** Engineering measures and benchmarking information of Pincers B, C, D, and E

	Engineering measures in HoQ					Benchmarking information in HoQ				
	ECs	EC1	EC2	EC3	EC4	CR <sub>1</sub>	CR <sub>2</sub>	CR <sub>3</sub>	CR <sub>4</sub>	Satisf
	(Units)	(mm)	(mm)	(mm)	( $\mu$ m)					(%)
Pincers B	Ours	225	960	24	40	2.8	3.5	4.1	2.5	54.3
	Co1	238	990	28	20	4.6	2.6	3.4	2.5	59.4
	Co2	235	985	27	30	4.2	3.1	3.5	2.2	59.3
	Co3	232	975	26	60	3.8	4.6	3.7	1.1	62.2
	Co4	228	965	25	20	3.2	2.5	3.9	3.2	52.6
	Min	220	950	20	20					
	Max	260	1,050	40	60					
Pincers C	Ours	305	1,060	24	40	2.3	3.5	4.1	3	52.4
	Co1	318	1,090	28	20	4.3	2.6	3.4	3	59.5
	Co2	315	1,085	27	30	3.9	3.1	3.5	2.7	59.1
	Co3	312	1,075	26	60	3.4	4.7	3.7	1.6	61.5
	Co4	308	1,065	25	20	2.7	2.5	3.9	3.7	51.1
	Min	300	1,050	20	20					
	Max	340	1,150	40	60					
Pincers D	Ours	345	1,160	36	40	2.7	3.6	3.9	2.8	54.2
	Co1	358	1,190	32	20	4.7	2.7	3.2	2.9	62.5
	Co2	355	1,185	33	30	4.3	3.1	3.3	2.6	61.1
	Co3	352	1,175	34	60	3.8	4.3	3.5	1.5	60.4
	Co4	348	1,165	35	20	3.1	2.8	3.7	3.6	55
	Min	340	1,250	20	20					
	Max	380	1,250	40	60					
Pincers E	Ours	385	1,260	24	40	2.8	3.1	4.1	3.4	54.4
	Co1	398	1,290	28	20	4.5	2.6	3.4	3.1	61.8
	Co2	395	1,285	27	30	4.2	2.8	3.5	3	60.8
	Co3	392	1,275	26	60	3.7	3.7	3.7	2.3	60.6
	Co4	388	1,265	25	20	3.1	2.5	3.9	3.8	54.6
	Min	380	1,250	20	20					
	Max	420	1,350	40	60					

As shown in Table 14.3, the optimal values of EC3 of pincers A, B, C, D, and E are the same. Apparently, they can be grouped into one group. On the other hand, the values of EC4 are different from each other and clustering analysis is required. The result is that the values of EC4 are clustered into two groups: one group includes pincers A and B and the other includes pincers C, D, and E. The corresponding optimal values of the platform and non-platform ECs are listed in Table 14.5.

In the computation process, many optimization models are required to be established as described in Sect. 14.3. To help the reader better understand the modeling, the ILOG CPLEX code corresponding to the model in Sect. 14.3.4.1 and result of Table 14.5 is given as an example as follows:

**Table 14.2** The calculated coefficients for the pincers

Product	EC	Intercept	x1	x2	x3	x4
Pincers (A–E)	x1	0.04		1.02		
	x2	−0.03	0.98			
Pincers A	y1	2.02	2.67	3.25		
	y2	2.37	−0.08		0.56	2.13
	y3	4.32	−0.9	−1.3		
	y4	3.84	−1.1	−1.85	−0.38	−2.78
Pincers B	y1	2.22	2.37	3.25		
	y2	2.37	−0.08		0.56	2.13
	y3	4.32	−0.9	−1.3		
	y4	3.84	−1.1	−1.85	−0.38	−1.82
Pincers C	y1	1.62	2.67	3.75		
	y2	2.37	−0.08		0.56	2.16
	y3	4.32	−0.9	−1.3		
	y4	4.34	−1.1	−1.85	−0.38	−1.82
Pincers D	y1	2.05	2.37	3.93		
	y2	2.37	−0.08		0.56	1.53
	y3	4.12	−0.9	−1.3		
	y4	4.34	−1.1	−1.85	−0.38	−1.82
Pincers E	y1	2.12	2.87	2.75		
	y2	2.37	−0.08		0.56	1.23
	y3	4.32	−0.9	−1.3		
	y4	4.34	−1.1	−1.55	−0.38	−1.22

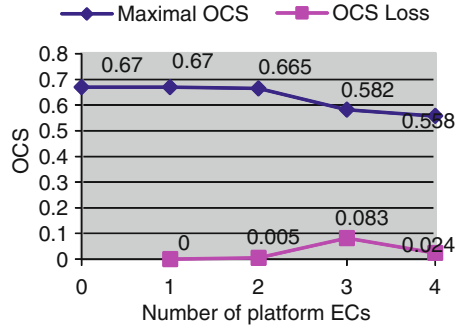
**Table 14.3** The optimal values of the ECs for the product profiles

Product	EC1	EC2	EC3	EC4	Satisf. (%)
A	201.10	898.29	40.00	34.29	63.2
B	240.91	997.34	40.00	42.27	66.4
C	322.12	1100.81	40.00	51.10	69.8
D	359.84	1195.20	40.00	54.75	67.0
E	401.35	1298.93	40.00	60.00	68.6
Average					67.0

**Table 14.4** The calculated sensitivity indices of ECs

Sensitivity index (SI)		EC1	EC2	EC3	EC4
Local SI	Product A	0.139	0.143	0.023	0.021
	Product B	0.112	0.143	0.023	0.069
	Product C	0.139	0.187	0.023	0.071
	Product D	0.112	0.203	0.023	0.024
	Product E	0.156	0.115	0.023	0.031
Global SI		0.132	0.158	0.023	0.043

**Fig. 14.4** The maximal OCS and OCS loss with different number of platform ECs



**Table 14.5** The optimal solution achieved by the proposed approach

Product	EC1	EC2	EC3	EC4	Satisf. (%)
A	201.10	898.29	40.00	34.29	63.2
B	240.91	997.34	40.00	34.29	65.0
C	322.12	1100.81	40.00	51.10	69.8
D	359.84	1195.20	40.00	51.10	66.7
E	401.35	1298.93	40.00	51.10	67.9
Average					66.5

```
//ILOG CPLEX code:
int nCRs=...;//number of CRs;
int nECs=...;//number of ECs;
int nPPs=...;//number of product profiles

range rCRs=1..nCRs;
range rECs=1..nECs;
range rRegEffis=1..nECs+1;//coefficients of the regression
formulation (plus constant column)
range rPPs=1..nPPs;

float w[rPPs][rCRs]=...;//weights of CRs
float y_max=...;//maximal value of y;
float y_min=...;//minimal value of y;

float fRelaECs[rPPs][rECs][rRegEffis]=...;//co-relationship
of ECs
float fRelaCRs2ECs[rPPs][rCRs][rRegEffis]=...;//relationship
between CRs to ECs

dvar float+x[rPPs][rECs];;//normalized target value of ECs
dvar float+y[rPPs][rCRs];;//customer perception of the degree
of achievement of CRs
dvar float+x_h[rECs];;//shared value of ECs
```

```

maximize//objective function: instantiation of Eq. (14.7)
  sum(p in rPPs)
    sum (i in rCRs)
      w[p][i]*((y[p][i]-y_min)/(y_max-y_min)/nPPs);

constraint c1;//relationship between CRs and ECs: instan-
tiation of Eq. (14.8)
constraint c2;//corelationship among ECs: instantiation
of Eq. (14.9)
constraint c3;//shared EC constraint: instantiation of Eq.
(14.10)
constraint c4;//shared EC constraint: instantiation of Eq.
(14.10)
constraint c5;//shared EC constraint: instantiation of Eq.
(14.10)
constraint c6;//confine the bound of y
constraint c7;//confine the bound of x

subject to
{
  c1=forall (p in rPPs)
    forall (i in rCRs)
      y[p][i]==fRelaCRs2ECs[p][i][1]+sum(j in 2..nECs
+1) (x[p][j-1]*fRelaCRs2ECs[p][i][j]);
  c2=forall (p in rPPs)
    forall (j in rECs:j<=2)
      x[p][j]==fRelaECs[p][j][1]+sum (j1 in 2..nECs
+1) (x[p][j1-1]*fRelaECs[p][j][j1]);
  c3=forall (p in rPPs)
    x[p][3]==x_h[1];
  c4=forall (p in rPPs:p<=2)
    x[p][4]==x_h[2];
  c5=forall (p in rPPs:p>2)
    x[p][4]==x_h[3];
  c6=forall (p in rPPs)
    forall (i in rCRs)
    {
      y[p][i]>=1.0;
      y[p][i]<=5.0;
    }
  c7=forall (p in rPPs)
    forall (j in rECs)
      x[p][j]<=1.0;
}

```

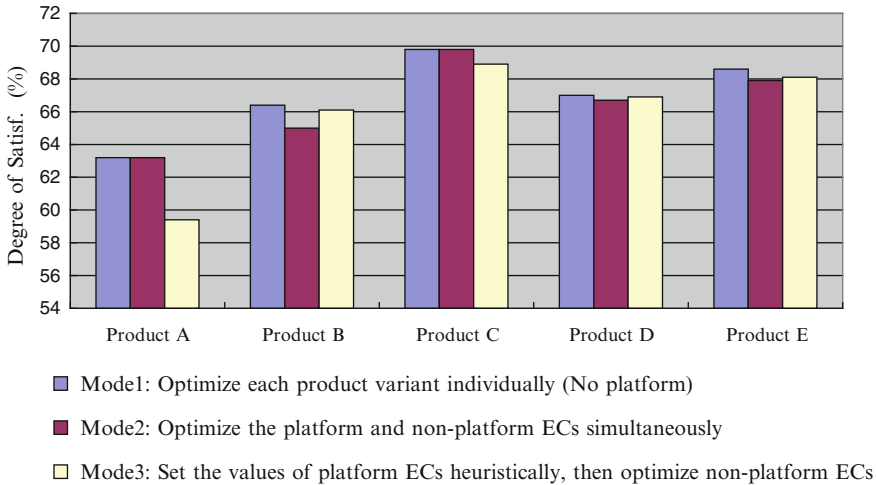


Fig. 14.5 Comparison of Mode 1, Mode 2, and Mode 3

One can see from Table 14.3 that the average degree of customer satisfaction to all product profiles is 67.0 % if the optimal ECs are obtained individually for each product profiles. However, when the values of platform ECs are confined to the shared values accordingly, the average degree of customer satisfaction is declined to 66.5 % as shown in Table 14.5. This indicates that 0.5 % degree of customer satisfaction is lost due to the commonization of the product platform. Figure 14.5 depicts the comparison result of these two different modes on degree of customer satisfaction for each pincers. The loss of customer satisfaction is mainly caused by the pincers B, D, and E because their ECs are forced to change to the shared values.

### 14.4.3 Advantages Over a Two-Stage Approach

Some product platform design approaches use two-stage optimization. For example, the values of platform ECs are determined heuristically, and then the non-platform ECs of each product profile are optimally instantiated. However, two-stage approaches may not produce the best overall performance of a product family because the optimization is partitioned into two or more stages (Messac et al. 2002). For comparison, we calculated the platform ECs heuristically according to the weighed-sum method provided by Dai and Scott (2007) as follows:

$$\hat{x}_{jl} = \left( \sum_{k' \in V_{j1}} x_j^{k'} * SI_j^{k'} \right) / \sum_{k' \in V_{j1}} SI_j^{k'}, (l = 1, 2, \dots, n_j, j' \in U) \quad (14.18)$$



**Table 14.6** The optimal solution achieved by a two-stage approach

Product	EC1	EC2	EC3	EC4	Satisf. (%)
A	195.23	883.91	40.00	40.43	59.4
B	240.91	997.34	40.00	40.43	66.1
C	320.32	1096.38	40.00	53.98	68.9
D	359.84	1195.20	40.00	53.98	66.9
E	401.35	1298.93	40.00	53.98	68.1
Average					65.9

where  $SI_j^{k'}$  is the local SI of the  $j'$  th EC of the  $k'$  th product profile. By applying Eq. (14.18) with the optimal values of ECs in Table 14.3 and the local SIs in Table 14.4, the common value of the EC3 is calculated as 40.0 and the common values of EC4 for the first and the second group are calculated as 40.43 and 53.98, respectively. Then the obtained platform ECs are set as given parameters, and the optimal non-platform ECs for each product profile are obtained by solving the optimization model. The result of this approach is listed in Table 14.6. One can see that the average degree of customer satisfaction of this approach is 65.9 %, which is 0.96 % lower than that of the single-stage approach. A detailed comparison on each product profile between these two approaches is also given in Fig. 14.5. The comparison results show that, based on the data of this case, simultaneously optimizing platform and non-platform ECs yields better solutions although it may cause a very few increase on computation time.

## 14.5 Discussions and Conclusions

Determination of the optimal target values for the platform and non-platform ECs is an important activity in product platform design process. In this paper, a five-step QFD-based optimization method is presented with a view to achieving a high level of overall customer satisfaction. By using the method proposed in this paper, the optimal solution for the problem can be achieved with the maximal average degree of customer satisfaction of all product profiles, and thus the total quality loss of the product family due to commonization is minimized.

In addition, our approach can be categorized as a kind of single-stage optimization approach, which seeks to optimize product platform and corresponding family of products simultaneously (Simpson 2004). Compared with two-stage approaches, single-stage approach expectedly yields improvements in the overall performance of the product family (Messac et al. 2002). The comparison results based on the case data in Sect. 14.4.3 also show that our approach has a higher average degree of customer satisfaction. The main disadvantage of a single-stage approach is the higher dimensionality of the optimization formulation. Nevertheless, the optimization model in proposed approach is a linear programming one; thus the computation time is tolerable.

One limitation of the proposed approach is that for large-scale problem in which a large quantity of product profiles are involved, the quantum of the work for developing HoQs of product profiles and performing the regression may increase considerably. Therefore, an interactive software package is suggested to be developed to automatically establish HoQs, perform parameters regression, and generate SIs for large-scale applications. Although in theory the proposed method can be used for general-type scalable product platform, experiments for very complex products or systems are not performed so far. Therefore, the applicability of the propose method for very complex products or systems is not verified.

Another limitation is that in this research we only consider the linear forms of  $f$ ,  $g$ , and  $h$  in the optimization models. However, they can be in complex nonlinear forms in practical scenario. One of our research potentials is to build these nonlinear relationships by using appropriate approximation methods based on existing product data and to establish corresponding optimization models.

**Acknowledgments** This research was financially supported by the National Science Foundation of China (NSFC Proj. 71171039, 61273204, and 71021061) and the Fundamental Research Funds for Central Universities (Proj. N110204005).

## References

- Akao Y (1990) Quality function deployment: integrating customer requirements into product design. Productivity Press, Cambridge, MA
- Chen Y, Tang J, Fung RYK, Ren Z (2004) Fuzzy regression-based mathematical programming model for quality function deployment. *Int J Prod Res* 42:1009–1027
- Chen Y, Fung RYK, Tang J (2005) Fuzzy expected value modelling approach for determining target values of engineering characteristics in QFD. *Int J Prod Res* 43:3583–3604
- Cohen L (1995) Quality function deployment: how to make QFD work for you. Addison-Wesley, New York, NY
- Conner C, de Kroon JP, Mistree F (1999) A product variety tradeoff evaluation method for a family of cordless drill transmissions. In: 1999 ASME design technical conference, Las Vegas, Nevada, Paper No. DETC99/DAC-8625
- Dai ZH, Scott MJ (2007) Product platform design through sensitivity analysis and cluster analysis. *J Intell Manuf* 18:97–113
- Erixon G, Von YA, Arnstrom A (1996) Modularity: the basis for product and factory reengineering. *Ann CIRP* 45:1–4
- Fellini R, Kokkolaras M, Michelena N, Papalambros P, Saitou K, Perez-Duarte A, Fenyes P (2004) A sensitivity based commonality strategy for family products of mild variation, with application to automotive body structures. *Struct Multidiscip Optim* 27:89–96
- Fujita K (2002) Product variety optimization under modular architecture. *Comput Aided Des* 34:953–965
- Fung RYK, Popplewell K, Xie J (1998) An intelligent hybrid system for customer requirements analysis and product attribute targets determination. *Int J Prod Res* 36:13–34
- Fung RYK, Tang JF, Tu YL, Wang DW (2002) Product design resources optimization using a non-linear fuzzy quality function deployment model. *Int J Prod Res* 40:585–599
- Hauser JR, Clausing D (1988) The house of quality. *Harv Bus Rev* 66:63–73
- Hsiao SW, Liu E (2005) A structural component-based approach for designing product family. *Comput Ind* 56:13–28

- Jariri F, Zegordi SH (2007) Quality function deployment planning for platform design. *Int J Adv Manuf Technol* 36:419–430. doi:[10.1007/s00170-006-0853-3](https://doi.org/10.1007/s00170-006-0853-3)
- Jiao JX, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18:5–29
- Karsak EE, Sozer S, Emre AS (2003) Product planning in quality function deployment using a combined analytic network process and goal programming approach. *Comput Ind Eng* 44:171–190
- Khoo LP, Chen CH, Yan W (2002) An investigation on a prototype customer-oriented information system for product concept development. *Comput Ind* 49:157–174
- Kim KJ, Moskowitz H, Dhingra A, Evans G (2000) Fuzzy multicriteria models for quality function deployment. *Eur J Oper Res* 121:504–518
- Kim KJ, Lee DU, Lee MS (2006) Determining product platform elements for mass customization. *Int J Prod Qual Manag* 1:168–182
- Kumar D, Chen W, Simpson TW (2006) A market-driven approach to the design of platform-based product families. In: 11th AIAA/ISSMO multidisciplinary analysis and optimization conference 200–224, Portsmouth, VA
- Lin MC, Wang CC, Chen MS, Chang CA (2008) Using AHP and TOPSIS approaches in customer-driven product design process. *Comput Ind* 59:17–31
- Madu CN (2006) *House of quality (QFD) in a minute*. Chi Publishers, Fairfield, CT
- Martin MV, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. *Res Eng Des* 13:213–235
- Messac A, Martinez MP, Simpson TW (2002) Effective product family design using physical programming. *Eng Optim* 34:245–261
- Meyer MH, Lehnerd AP (1997) *The power of product platforms: building value and cost leadership*. Free Press, New York, NY
- Meyer MH, Utterback JM (1993) The product family and the dynamics of core capability. *Sloan Manage Rev* 34:29–47
- Moore WL, Louviere JJ, Verma R (1999) Using conjoint analysis to help design product platforms. *J Prod Innov Manag* 16:27–39
- Moskowitz H, Kim KJ (1997) QFD optimizer: a novice friendly quality function deployment decision support system for optimizing product design. *Comput Ind Eng* 33:641–655
- Nayak RU, Chen W, Simpson TW (2002) A variation-based method for product family design. *Eng Optim* 34:65–81
- Park T, Kim KJ (1998) Determination of an optimal set of design requirements using house of quality. *J Oper Manag* 16:469–581
- Simpson TW (2004) Product platform design and customization: status and promise, artificial intelligence for engineering design. *Anal Manuf* 18:3–20
- Simpson TW, Maier JRA, Mistree F (1999) A product platform concept exploration method for product family design. Design theory and methodology – DTM'99, Las Vegas, Nevada, ASME, Paper No.DETC99/DTM-8761
- Simpson TW, Siddique Z, Jiao RJ (2006) *Product platform and product family design: methods and applications*. Springer, New York, NY
- Tang JF, Fung RYK, Xu BD, Wang DW (2002) A new approach to quality function deployment planning with financial consideration. *Comput Oper Res* 29:1447–1463
- Thevenot HJ, Simpson TW (2006) Commonality indices for product family design—a detailed comparison. *J Eng Des* 17:99–119
- Wang J (1999) Fuzzy outranking approach to prioritize design requirements in quality function deployment. *Int J Prod Res* 37:899–916
- Wassermann GS (1993) On how to prioritize design requirements during the QFD planning process. *IIE Trans* 25:59–65
- Williams CB, Allen JK, Rosen DW, Mistree F (2007) Designing platforms for customizable products in markets with non-uniform demand. *Concurrent Eng: Res Appl* 15:106–201

# Chapter 15

## Cascading Platforms for Product Family Design

Jiju A. Ninan and Zahed Siddique

**Abstract** Product family design is a trade-off between distinctiveness of products in the family and commonality between them. Increasing the commonality of components can lead to loss of performance of product variants. Saving in cost comes at the expense of performance of products. Therefore selection of components to be standardized across the family and their configuration is a critical step in the design of product families. A common approach to the product family design is to treat it as a design optimization problem so that trade-off decisions between commonality and performance can be performed. In this chapter we present a scale-based multi-platform optimization approach. The approach uses systematic relaxation to increase leverage among multiple platforms and provide increase in performance for family members supported by the platform. The three stages involved in the approach are (1) single platform, (2) platform evaluation, and (3) platform relaxation. The Black and Decker universal motor family is used to demonstrate the approach.

### 15.1 Introduction

Product development enterprises normally offer a range of products varying from low cost-low performance to high cost-high performance products to serve different market segments. Traditionally, the product varieties were individually designed and manufactured to suit the requirements of the particular market segment.

---

J.A. Ninan  
Schlumberger, Sugarland, TX, USA

Z. Siddique (✉)  
School of Aerospace and Mechanical Engineering, University  
of Oklahoma, Norman, OK 73019, USA  
e-mail: [zsiddique@ou.edu](mailto:zsiddique@ou.edu)

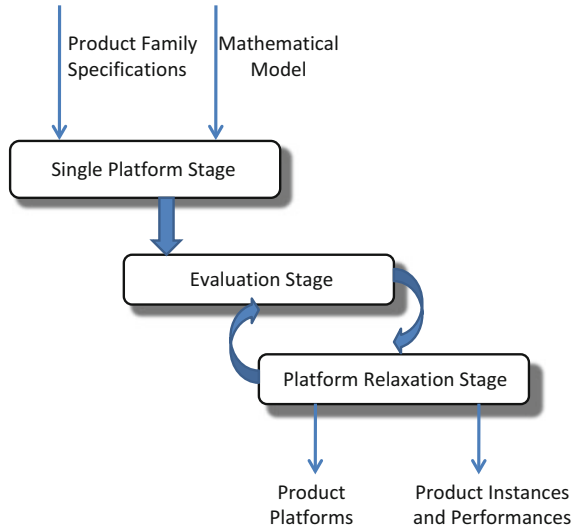
Each product, in a family, had different components, even though they served the same or similar functions. The products in a family lacked commonality among the different products, resulting in high cost in design, manufacturing, and inventory. These costs can be reduced or eliminated by sharing components and parts among the different family members. Companies are moving towards using platforms to support families, as a result of these advantages. Based on the product differentiating factors, product families can be classified as (1) modular product families—wherein product family members are instantiated by adding, substituting, and/or removing one or more functional modules from the product platform and (2) scalable product families—wherein scaling variables are used to “stretch” or “shrink” the product platform in one or more dimensions to obtain the different product variants.

The trade-off between performance and commonality is one of the main concerns in determining a product platform for a set of similar products. Designers need to balance the commonality of the products in the family with the individual distinctiveness of each product in the family. Normally a product family design process includes (1) designing the platform and (2) designing the individual product variants from the platform. Therefore, product family design should focus on the design of the entire family and platform, as well as the individual products. Several researchers have treated the design of product families as a design optimization problem (see Sect. 15.2). The advantage of this methodology is that designers can maintain a balance between commonality and cost. The platform and family members can simultaneously be optimized for performance, cost, and commonality during product design.

Based on the number of stages involved in the design process, product family optimization methods can be categorized as single-stage and multistage design optimization. Single-stage approaches seek to simultaneously optimize the product platform and corresponding products in the family of, while multistage approaches optimize the platform first and then instantiate the individual products from the platform. Single-stage optimization usually requires only one optimization run, but the size of the optimization problem increases tremendously as the number of parameters and number of products in the family increases. Multistage optimization breaks the larger problem into smaller subproblems. They require at least one optimization run for determination of the platform, and “ $n$ ” optimization runs to leverage the “ $n$ ” products in the family from the platform.

A product family can be supported using a single platform, which may cause poor performance of individual product family members. Hence, a single platform might not be sufficient to successfully leverage all of the products in the family (Dai and Scott 2007). In multi-platform, the products are leveraged from two or more platforms so that the loss of performance due to commonalization can be reduced. In multi-platform design, the challenges are to find (1) the minimum number of platforms that can serve the family of products with minimal loss of performance, (2) the platform from which each product is leveraged from, and (3) which parameters constitute the platform parameters for each platform. In a multi-platform approach, one of the drawback is that the platforms can be

**Fig. 15.1** Platform relaxation method inputs and outputs



unrelated, increasing the cost in design, manufacturing, assembly, etc. Hence, a multi-platform approach that explicitly tries to ensure some level of commonality among platforms can be more efficient. The approach presented in this chapter combines the single and multi-platform approach to systematically relax a single platform into multi-platforms, which are related through a set of common parameters, to increase performance of the family. The systematic relaxation of parameters allows different platforms to maintain commonality among them.

The inputs to the relaxation formulation are the parametric description of the products, constraints related to performance of products, and underlying mathematical models relating the product parameters to constraints and objectives. The proposed approach consists of three stages (Fig. 15.1):

1. Single platform stage
2. Platform evaluation stage
3. Platform relaxation stage

The starting point for the proposed approach is a product family based on a single platform optimization. The products are designed assuming that a single platform is sufficient to scale all the products in the family. In the second stage, the resulting product family from the first stage is evaluated. The loss of performance due to commonality for each product or the family as a whole is evaluated. Family members, optimized individually for performance, are used as the benchmark for comparison. In the platform relaxation stage, the single platform parameters are systematically relaxed and explored to increase performance of selected family members. The objective of systematic platform parameter relaxation ensures commonality among the first and the succeeding platforms while generating improved products.

## 15.2 Literature Review

Several researchers have used optimization approaches to design family of products to arrive at a suitable a product platform and also product varieties. Optimization approaches are used to perform trade-offs between commonality (the underlying platform) and the performance of the product variants. Researchers have studied the effect of commonality on individual product performances such as cost, efficiency, strength, and reliability of the product variants. A comprehensive review and classification of product family optimization methods was presented by Simpson (2006). Belloni et al. (2008) compared heuristic optimization approaches for product line design problems. Some of the product platform and family optimization approaches are highlighted next.

Simpson (1998) developed the Product Platform Concept Exploration Method (PPCEM), which is a multistage method for design of scale-based product families. The formulation returns the optimized product platform and the product family instances. Variation-Based Methodology for Product Family Design (VBPDM) (Nayak et al. 2002) is an extension to PPCEM. In VBPDM the platform and scale variables are identified in the optimization formulation. Nelson et al. (2001) presented a multi-criteria optimization model to determine trade-off decisions in product family designs. The authors demonstrated how to generate the Pareto set in case of product family design, where each product family member has different objective functions. Product Family Penalty Function using Physical Programming (PFPF) developed by Messac et al. (2002) uses physical programming (Messac 1996) for product family design. The difference between PFPF and PPCEM is that in PPCEM, the scale variables of the product family needs to be known prior.

Dai and Scott (2007) presented a method for product family design using cluster analysis and sensitivity analysis. They presented a multi-platform design method, where design variables may be shared among variants using any possible combination of subsets. A drawback of the suggested method is that as the problem size increases, the complexity of the problem will also tremendously increase. Simpson and D'Souza (2004) presented a product family design approach using genetic algorithm to simultaneously design the family of products while considering varying levels of commonality within the product family.

Fellini et al. (2006) proposed an approach for obtaining desired level of commonality through problem size reduction in platform selection and then maximizing commonality among variants, while minimizing individual performance deviations. Khire et al. (2006) presented a method to specify the platform configuration and optimize the design of the platform and the individual variants by choosing design variable values while maintaining commonality defined in the platform configuration. Khajavirad and Michalek (2007) presented a decomposed gradient-based approach to jointly determine the optimal selection of components to be shared across product variants and the optimal values for design variables that define those components. Moon et al. (2011) presents a multi-objective particle swarm optimization (MOPSO) approach to select the best platform design strategy

from a set of Pareto-optimal solutions based on commonality and design variation within the product family. Gao et al. (2009) presented a three-step approach for finding the optimal values of platform and non-platform parameters in a modular scale-based platform.

### 15.3 Platform Relaxation for Multi-platform Design: Overall Approach

The platform relaxation method is a three-stage design process (Fig. 15.1). The first stage is to identify a common platform that can support the entire family of products (single platform stage). The performance of the family members is then evaluated by comparing them with the benchmarks using predetermined criteria. Benchmark products are optimized individually (without commonality) with the same specification as the corresponding family member. Family members that perform inadequately, compared to the benchmark, are segregated and separated out from the current platform. This stage is referred to as the evaluation stage. In the last stage, the parameters of the initial platform are relaxed systematically to arrive at a new platform to support the set of products separated out. Relaxing or cascading the platforms helps to attain commonality between different platforms to achieve higher cost savings. The resulting products, supported by the relaxed platform, are again evaluated and the platform is again cascaded if necessary. The design process is iterative and is continued until all members of the product family have acceptable performance.

The specifications of the product family members and the underlying mathematical model are provided by the designer. The mathematical model usually comprises of bounds and constraints on the design parameters and parametric relation among design parameters and responses. The platform relaxation method returns the parameter values of product family members, configuration of different platforms, platform from which each product is leveraged, performance of product family members, and their performance loss due to commonality. In scale-based product family architecture (Table 15.1), each product instance  $P_1, P_2, \dots, P_m$  of the family can be uniquely and completely described by the same set of product parameters  $x_1, x_2, \dots, x_n$ . These parameters describe the attributes of the physical components present in the products. If the value of any of the parameters is constant throughout the family (in case of single platform) or a subset of products (in case of multi-platform), the parameter is said to be a platform parameter. The product parameters related to the entire product family are defined as  $x_{ij}$ , which indicates the  $i$ th parameter for the  $j$ th product. The entire family of products is then a set of design parameters ( $x$ ). The design task is to find the value of the parameters in  $x$  that results in a product family with maximum commonality and minimal loss in performance.  $Y$  is the set of platform commonality variables corresponding to the product parameters  $X$ .  $PP$  is the set of product platforms used to leverage the products.



**Table 15.1** General steps in platform relaxation method

---

$PF = \{p_1, p_2, \dots, p_m\}$   
 $Y = \{y_1, y_2, \dots, y_n\}$   
 $PP = \{pp_k \mid pp_k \text{ is the set of product platforms from which the family is derived}\}$   
 $PP = \{pp_1, pp_2, \dots, pp_f\}$   
 $P_{ck} = \{pp_{ck} \mid pp_{ck} \text{ is the set of products considered for leveraging from the platform "k"}\}$   
 $X_{pk} = \{x_{pk} \mid x_{pk} \text{ is the set of platform parameters for platform "k"}\}$   
 $C_{ik}$  is the set of platform parameter values in platform "k" (if  $i \in x_{pk}$ )  
 $N_k = \text{Cardinality of } x_{pk}$

**1. Single platform stage**

1. Execute "individual optimization formulation"
2.  $k = 1$
3.  $P_{c1} = PF, P_{c1} = \emptyset$
4. Execute "platform-specified/non-platform-specified formulation"
5.  $k = k + 1$

**2. Evaluation stage**

6.  $\Delta_j = f_{\text{benchmark},j}(w_1z_1 + w_2z_2 + \dots + w_nz_n) - f_{\text{family},j}(w_1z_1 + w_2z_2 + \dots + w_nz_n) \forall j \in P_{ck-1}$   
 Case 1: All  $\Delta_j$  values  $< \eta$ ;  $P_{ck} = \{ \}$ ; Goto 9  
 Case 2: Some  $\Delta_j$  values  $< \eta$  & other  $\Delta_j$  values  $> \eta$  then (a)/(b)  
     (a) Include products with  $\Delta_j$  values  $> \eta$  in  $P_{ck+1}$ ;  $k = k + 1$ ; Goto 7  
     (b) Include all products in  $P_{ck}$ ; Goto 7  
 Case 3: No  $\Delta_j$  values  $< \eta$ ;  $P_{ck} = P_{ck-1}$ ; Goto 7

**3. Cascading stage**

7. Execute "Platform Relaxation Formulation"
8. Goto 6;
9. End

---

Initially the number of platforms required is unknown.  $X_{pk}$  is the set of variables for each platform.  $N_k$  represents the number of platform parameters in each platform. The steps associated with the platform relaxation method and associated optimization formulations are discussed in the following subsections.

### 15.3.1 Single Platform Stage

The starting point for the platform relaxation method is determining the entire product family using a single platform, where platform parameters have the same value for all the products in the family. There are two possible formulations that have been widely used to determine the single platform (1) platform specified and (2) non-platform specified. In platform-specified approach, the designer specifies the platform parameters. The aim of the optimization formulation is to arrive at an optimum  $x (\hat{x})$  which enforces commonality of specified platform parameters throughout the family and also minimizes the loss of performance due to commonality. In the non-platform-specified formulation (Table 15.2), the aim is to explore different levels of commonality and perform trade-off between commonality and

**Table 15.2** General formulation for non-platform specified optimization

---

**Indices**

$j$  = Product family members,  $j \in J, J = \{1,2,3, \dots, m\}$   
 $t$  = Product Constraints,  $t \in T, T = \{1,2,3, \dots, s\}$   
 $l$  = System goals,  $l \in L, L = \{1,2,3, \dots, p\}$

**Variables**

$x_{ij}$  is the parameter ‘i’ in product ‘j’  
 $y_1, y_2, \dots, y_n$  are the commonality parameters corresponding to each parameter in  $I$   
 $G_{1g}$  is the target goal of objective 1 for product j  
 $d^+_{1j}$  is the positive deviation of the first goal for jth product  
 $d^-_{1j}$  is the negative deviation of the first goal for jth product  
 $w_{1j}$  is the weights for the deviation variables  $d_{1j}^{+/-}$  in the objective function  
 $w_i$  is the weights for the commonality parameters in the objective function

**Objective**

$$\sum_{l=1}^p \sum_{j=1}^m f(w_{lj}, d^-_{lj}, d^+_{lj}) - \sum_{i=1}^n w_i y_i$$

**Subject to**

$x_{ij}^{lower} \leq x_{ij} \leq x_{ij}^{upper}, \forall i \in I$  and  $\forall j \in J$  Bounds on the design variable  
 $y_i = \begin{cases} 1 & \text{when } x_i \text{ is a platform} \\ 0 & \text{when } x_i \text{ is not a platform} \end{cases}$   
 $g_t(x) = 0, t = 1, 2, \dots, s$  Constraint relating to individual products  
 $(x_{ij} - x_{ij+1})y_i = 0 \forall i \in I$  and  $j \in J, j \neq m$  Commonality constraints  
 $y_i^2 - y_i = 0 \forall i \in I$  Constraints for converting  $y_i$  to continuous variables  
 $A_{lj}(x) + d^-_{lj} + d^+_{lj} = G_{lj} \forall l \in L$  and  $j \in J$ , Objectives transformed to system goals  
 $d^-_{lj}, d^+_{lj} \geq 0, d^-_{lj} * d^+_{lj} = 0 \forall l \in L$  and  $j \in J$ , Non negativity of deviation variables

---

loss of performance of family members to arrive at a suitable product platform and leverage the product family members using the platform.

In both instances, platform commonality can be modeled mathematically for the family by

$$x_{ij} = x_{ij+1} j, j \neq m \text{ if } i \in x_p \tag{15.1}$$

where  $x_p$  is the set of platform variables. As mentioned earlier to represent sharing of parameters, a set of binary decision variable (0, 1) corresponding to each product parameter are utilized. These platform commonality decision parameters are represented by  $y_i$ .

$$y_i = \begin{cases} 1 & \text{when the parameter is a platform parameter} \\ 0 & \text{when the parameter is a scale parameter} \end{cases}$$

$y_i$  parameters can be used to turn on/off the commonality of corresponding parameters. In the platform-specified formulation,  $y_i$  values of platform parameters are set to 1 to enforce commonality. In non-platform-specified formulation,

$y_i$  helps to explore levels of commonality by activating or deactivating platform commonality for different parameters. The commonality is maximized through trade-offs between maximum number of platform parameters and performance loss for family members. In this chapter we use the non-platform-specified design formulation. The general form of the non-platform-specified formulation is shown in Table 15.2.  $g(X)$  are the constraints that need to be satisfied by each product;  $l_i$  and  $u_i$  represent the lower and upper bounds of the product parameters  $x_i$ . The number of platform counter “ $k$ ” is set to 1 for the single platform stage. In the single platform stage,  $P_{cI}$  includes all the products in the family ( $P_{cI} = PF, P_{cI} = \emptyset$ ).

A goal programming model is adopted to consider multiple objectives for product family design. In goal programming the target values are identified for each objective, and the deviation of actual objective value from the targets is captured using deviation variables. Deviation variables  $d_{ij}^+$  and  $d_{ij}^-$  are the positive and negative deviation of actual attainment  $A_{ij}(x)$  from the target  $G_{ij}$ . Both  $d_{ij}^+$  and  $d_{ij}^-$  are constrained to have only nonnegative value.

$$\begin{aligned}
 &\text{If } A_{ij}(x) \leq G_{ij}(\text{underachievement}) \text{ then } d_{ij}^- > 0 \text{ and } d_{ij}^+ = 0 \\
 &\text{If } A_{ij}(x) \geq G_{ij}(\text{overachievement}) \text{ then } d_{ij}^+ > 0 \text{ and } d_{ij}^- = 0 \\
 &\text{and if } A_{ij}(x) = G_{ij}(\text{exactly satisfied}) \text{ then } d_{ij}^- = 0 \text{ and } d_{ij}^+ = 0
 \end{aligned}
 \tag{15.2}$$

When values larger than target are undesirable, the positive deviations are minimized in the objective function and vice versa. To keep the actual values close to target, both negative and positive deviations are minimized. The term  $\sum_{i=1}^n w_i y_i$  maximizes the number of platform parameters. Different terms in the objective function are weighted so that all of them are given equal priority while optimization is performed.

### 15.3.2 Evaluation Stage

The benchmarks for the evaluation stage are determined through optimizing the family members individually, subject to design, and performance requirements of corresponding product instance. The formulations (Table 15.3) are run independently, corresponding to each product in the family. The individual optimum corresponds to the best performance that can be achieved subject to requirements of the products.

In the evaluation stage products leveraged from the platform are compared against the benchmark products. Let  $z_1, z_2, \dots, z_p$  be the performance measures considered in the objective function;  $z_{1j}, z_{2j}, \dots, z_{pj}$  be their value for product “ $j$ ”; and  $z_{1j}^*, z_{2j}^*, \dots, z_{pj}^*$  be the value of their corresponding benchmark.

**Table 15.3** Individual optimization of product instances

Given:	Mathematical model Product constraints
Minimize	$f(w_1z_1, w_2z_2, \dots, w_nz_n)$
Subject to:	$g_r(X) \leq 0 \forall r \in R$ $l_i \leq x_i \leq u_i \forall i \in I$ (formulation repeated 'm' times for m products)

The performance of the products is  $\Delta_j = \pm (N_1z_{1j} - N_1z_{1j}^*) \pm (N_2z_{2j} - N_2z_{2j}^*) \dots \pm (N_pz_{pj} - N_pz_{pj}^*)$ . Here  $N_1, N_2, \dots, N_p$  are the factors used to normalize the performances for comparison. Depending on the nature of each desired performance measure, the following sign conventions are used for each of the factors in the function:

For positive valued targets: when performance higher than target is desired and the performance measure obtained for product “j” is higher than target, a negative sign is assigned, and when performance is lower than target, a positive sign is assigned.

For negative valued targets: when performance higher than target is desired and the performance measure obtained for product “j” is higher than target, a negative sign is assigned, and if the performance measure obtained for product “j” is lower than target, a negative sign is assigned.

When the performance measure is desired to be exactly equal to the target, a positive sign is assigned.  $\Delta_j$  values are calculated for each product leveraged from the current platform. After  $\Delta_j$  is calculated, the following cases represent the possible scenarios:

Case 1—All  $\Delta_j$  values  $< |\eta|$ : If all the products have performance within the acceptable limits,  $|\eta|$ , then further iterations or platforms are not required, and the platform relaxation method is considered to be complete.

Case 2—Some  $\Delta_j$  values  $< |\eta|$ : If some of the products satisfy the product performance criteria, while others do not, the designer has two possible options (a) include products with acceptable performance ( $\Delta_j$  values  $< |\eta|$ ) to be leveraged from the current platform, separate the nonconforming ( $\Delta_j$  values  $< |\eta|$ ) to be leveraged from relaxed platform. The platform count is now incremented by 1 to  $k = k + 1$ , and then the platform relaxation formulation is repeated with the nonconforming products. (b) Include both conforming and the nonconforming products and relaxed platform  $P_{ck-1}$  further. The choice between options (a) or (b) is dependent on the cost burden of developing another platform and the manufacturing processes involved.

Case 3—No  $\Delta_j$  values  $< |\eta|$ ;  $P_{ck} = P_{ck-j}$ : If none of the products are conforming, then the only option is to relax the platform further till conforming products are attained.

**Table 15.4** General platform relaxation formulation

Minimize	$\sum_{l=1}^p \sum_{j=1}^m f(w_{lj}, d_{lj}^-, d_{lj}^+)$
Subject to	$x_{ij}^{lower} \leq x_{ij} \leq x_{ij}^{upper}, \forall i \in I \text{ and } \forall j \in p_{ck}$ Bounds on the design variable $y_i = \begin{cases} 1 & \text{when } x_i \text{ is a platform} \\ 0 & \text{when } x_i \text{ is not a platform} \end{cases}$ $g_t(x) = 0, t = 1, 2, \dots, s$ Individual products constraints $(x_{ij} - C_{ik-1})y_i = 0 \forall i \in x_{pk-1} \text{ and } \forall j \in p_{ck}$ Commonality constraints $\sum y_i^3 \leq N_{k-1} - 1, \forall i \in x_{pk-1}$ Relaxation constraints $\sum y_i \geq N_{k-1} - 1, \forall i \in x_{pk-1}$ $A_{lj}(x) + d_{lj}^- + d_{lj}^+ = G_{lj} \forall l \in L \text{ and } j \in J$ , Objectives transformed to system goals $d_{lj}^-, d_{lj}^+ \geq 0, d_{lj}^- * d_{lj}^+ = 0 \forall l \in L \text{ and } j \in J$ , Non negativity of deviation variables

The threshold value influences the number of products that will be retained in the current platform. Changing the threshold value will change the platform leveraging approach. It is assumed that the designer specifies a reasonable threshold value for loss of performance.

### 15.3.3 Relaxation Stage

In this stage only the nonconforming products, separated during evaluation stage, is considered. Let  $p_{ck}$  be the set of products being considered for leveraging from the platform “ $k$ ” (Table 15.4). Let  $x_{pk}$  denote the platform parameters for the current platform  $pp_k$ . The idea is to arrive at a new platform  $pp_{k+1}$ , which consists of platform parameters  $x_{pk+1}$  formed by relaxing one of the platform parameters in  $x_{pk}$  to scalable parameter ( $x_{pk+1} \subseteq x_{pk}$ ). The value of platform parameters in  $x_{pk+1}$  is held same as that of  $x_{pk}$ .

The platform relaxation formulation starts from the previous platform, with all the platform parameters from the previous platform ( $pp_{k-1}$ ) held initially to the previous platform ( $C_{ik-1}$ ) values by applying the following constraints:

$$(x_{ij} - C_{ik-1})y_i = 0, \forall i \in x_{pk-1} \text{ and } \forall j = p_{ck} \tag{15.3}$$

Here  $C_{ik-1}$  corresponds to the value of platform parameters in the previous platform. Value of  $y_i$  parameters is held to 1 for  $C_{ik-1}$ . The objective of the formulation is to improve the performance of the products by relaxing the previous platform. The relaxation formulation is used to select a platform parameter that when converted to a scale parameter minimizes the deviation of performance. To achieve this two constraints are introduced:

$$\sum y_i^3 \leq N_{k-1} - 1, \forall i \in x_{pk-1} \tag{15.4}$$

$$\sum y_i \geq N_{k-1} - 1, \forall i \in x_{pk-1} \quad (15.5)$$

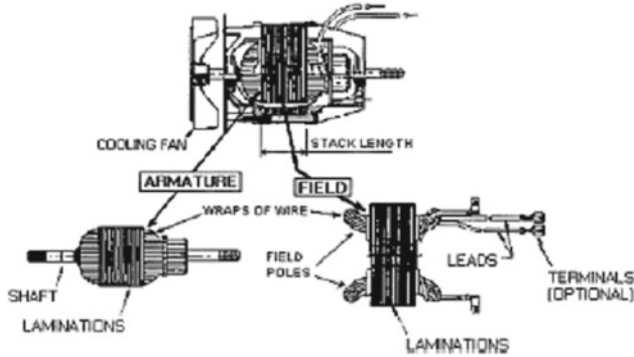
Here  $N_{k-1}$  is the number of platform parameters in the previous platform; the formulation selects one of the parameters that can be converted to scale parameters. To satisfy the above constraints, only  $(N_{k-1} - 1)$  number of  $y_i$  parameters are equal to 1, and the remaining  $y_i$  parameters have a value of 0. This constraint also ensures that  $y_i$  accepts only discrete, 0 or 1 value and no value in between. The objective function in this case is minimization of deviation parameters.

## 15.4 Universal Electric Motor Case Study

Universal electric motors are capable of operating on alternating current (AC) and direct current (DC). They deliver more torque for a given current than any other types of AC motors (Chapman 1991). The high-performance characteristics of the universal motor, coupled with their flexibility, have led to a wide variety of applications, especially in household appliances, where they are found in, e.g., electric drills and saws, blenders, vacuum cleaners, and sewing machines (Veinott and Martin 1986).

As shown in Fig. 15.2, a universal electric motor is composed of an armature and a field, which are also referred to as the motor and stator, respectively. The armature consists of a metal shaft and slats (armature poles) around which wire is wrapped longitudinally as many as 1,000 times. The field consists of a hollow cylinder within which the armature rotates. The field also has wire wrapped longitudinally around interior metal slats (field poles) as many as 100 times. In order to reduce cost, size, and weight, the motor that satisfies its performance requirements with the least overall mass and highest efficiency is considered to be the most desirable. The design objective is to design a family of ten universal electric motors that satisfy a variety of torque and power requirements and is supported using platform(s). Different varieties scaled from the platform(s) will meet specific requirements.

The product parameters for the electric motors are (1) number of turns in the armature, (2) number of turns in the field, (3) area of the armature, (4) area of the field wire, (5) radius of the motor, (6) thickness of the stator, (7) current drawn by the motor, and (8) stack length. The current is varied in each motor by using electrical resistors. There is no manufacturing advantage by holding current as a platform. Moreover, varying the current can help to achieve different power requirements without having to vary other parameters that affect the manufacturing process. Torque requirements for individual electric motors are  $T = \{0.05, 0.10, 0.125, 0.15, 0.30, 0.25, 0.30, 0.35, 0.40, \text{ and } 0.5\}$ . The constraint on magnetizing intensity ensures that the magnetic flux intensity within each motor does not exceed the physical flux carrying capacity of the steel. The constraint on feasible geometry ensures that the thickness of the stator does not exceed the radius of the stator since



<i>Name</i>	<i>Requirement</i>
Torque	$T = \{0.05, 0.10, 0.125, 0.15, 0.30, 0.25, 0.30, 0.35, 0.40, 0.5\}$
Power	$= 300 \text{ W}$
Magnetizing Intensity, H	$< 5000 \text{ A.turns/m}$
Feasible geometry	Radius of motor $>$ thickness of stator
Efficiency of each motor	$> 0.15$
Mass of each motor	$< 2.0 \text{ Kg}$

**Fig. 15.2** Requirements for the universal electric motor product family, adapted from Simpson et al. (2001)

the thickness is measured from the outside of the motor inward. The required output power is taken as 300 W, and the ten torques values range from 0.05 to 0.5.

There are two goals for each motor (1) efficiency and (2) mass, with targets of 79 % and 0.5 kg, respectively. A lower bound of 15 % on efficiency and an upper bound of 2.0 kg for mass are imposed for each product within the product family. The design requirements, range of possible values for product parameters, and the constraints related to the product family as introduced by Simpson et al. (2001) are shown in Table 15.4. In this case study, application of platform relaxation method for designing scale-based product families supported by multiple platforms is demonstrated. The platform relaxation method returns the configuration of the platform(s) from which each motor is leveraged, value of platform parameters, value of scale parameters for each motor, and performance of each motor.

The general steps of platform relaxation method, introduced in the previous section, are used for the design of electric motor product family. The three stages are explained in the following subsections. The general steps in the method are shown in Table 15.1. The product family *PF* consists of ten electric motors  $\{P_1, P_2, \dots, P_{10}\}$  with torque requirements of  $\{0.05, 0.10, 0.125, 0.15, 0.30, 0.25, 0.30, 0.35, 0.40, \text{ and } 0.5\}$  (Fig. 15.2). There are eight design parameters that describe

each product in the family; hence, there are eight platform commonality parameters in the set  $Y$ . These parameters are  $y_1, y_2, \dots, y_8$  corresponding to the product parameters  $x_1, x_2, \dots, x_8$ . The design objective is to find the optimum value of  $X$  for the product instances which results in minimum performance loss due to commonality, while maximizing commonality.

### 15.4.1 Single Platform Stage

In the single platform stage, all the products are considered for leveraging, hence  $P_{c1} = PF = P_1, P_2, \dots, P_{10}$ . The single platform optimization formulation is used to arrive at a platform that can be used to leverage all the products in the family. The single platform formulation for the universal motor family is shown in Table 15.5.

In this chapter, the universal electric motor case study is treated as a non-platform-specified case. In single platform optimization, a holistic view of the entire product family is adopted to determine a suitable single platform while simultaneously optimizing the platform and the product instances for maximum commonality and minimum loss of performance due to commonality. The objective function consists of minimizing the undesirable negative deviation of efficiency of each motor and positive deviation of mass for each motor and maximizing the number of platform parameters.

The platform commonality constraints are initially introduced as continuous variables ( $0 \leq y_i \leq 1$ ). Integerizing constraints are then used to force the formulation to accept only values of 0 or 1 (binary) for the  $y_i$  parameters. This allows the formulation to evaluate the model for values in between while moving to optimum values. This is required for the formulation to be implemented in a gradient-based optimization method. The commonality constraints are used to ensure that platform variables take the same value and scale variables take different value for different products in the family.

The model consists of 128 design variables and 180 constraints. The formulation was implemented in VRAND<sup>®</sup> Visual DOC<sup>®</sup>, a commercially available nonlinear optimization tool. Table 15.6 shows the results obtained from single platform optimization formulation. The formulation returned a platform consisting of parameters  $x_2, x_3, x_4, x_6, x_8$  with values of 70, 0, 38, 0.34, 5.91, and 1.62, respectively. Now the number of platform counter is incremented by a value of 1, but first the evaluation of products is performed.

### 15.4.2 Platform Evaluation Stage

During the single platform stage, all product family members are individual optimized to serve as benchmarks for evaluation. The results for individually optimized universal motors, subject to the requirements and considering no commonality between them, are shown in Table 15.7.



**Table 15.5** Single platform formulation applied to universal electric motor family*Variables* $x_{ij}$  = Product parameters in  $K$  for each family member  $j$  $y_j$  = Platform commonality variables $d_{Eff,j}^-$  = Negative deviation of Goal 1 (efficiency > 0.70) from the target for product  $j$  $d_{Eff,j}^+$  = Positive deviation of Goal 1 (Efficiency > 0.70) from the target for product  $j$  $d_{Mass,j}^-$  = Negative deviation of Goal 2 (efficiency > 0.50) from the target for product  $j$  $d_{Mass,j}^+$  = Positive deviation of Goal 2 (efficiency > 0.50) from the target for product  $j$  $E_j$  = Efficiency of motor  $j$   $M_j$  = Mass of motor  $j$ *Objective:*  $z = \sum_{j=1}^{10} d_{Eff,j}^- + \sum_{j=1}^{10} d_{Mass,j}^+ - \sum y_i$ *Subject to:*

(1) Bounds on the design variable

$$100 \leq x_{1j} \leq 1,500 \text{ turns} \quad 1 \leq x_{2j} \leq 500 \text{ turns} \quad 0.01 \leq x_{3j} \leq 1 \text{ mm}^2 \quad 0.01 \leq x_{4j} \leq 1 \text{ mm}^2$$

$$1 \leq x_{5j} \leq 1 \text{ cm} \quad 0.5 \leq x_{6j} \leq 10 \text{ mm} \quad 1 \leq x_{7j} \leq 6 \text{ amps} \quad 0 \leq x_{8j} \leq 10 \text{ cm}$$

$$1 \leq y_j \leq 1 \quad \text{where } \forall j \in J \text{ and } \forall i \in I$$

(2) Magnetic intensity of each motor less than 5,000 A turns/m

$$\frac{\pi(2x_{2j}x_{7j})}{\pi(2x_{5j}+x_{6j})/2+(2(2x_{5j}-x_{6j}-0.007)+2*0.007)} \leq 5000$$

(3) Feasible geometry:  $x_{6j} < x_{5j}$ 

(4) Mass of motor (M) &lt; 2.0 kg

$$\pi d_{steel} x_{7j} (x_{5j}^2 - (x_{5j} - x_{6j})^2) + \pi d_{steel} x_{7j} (x_{5j}^2 - (x_{5j} - x_{6j} - l_{gap})^2) (2x_{7j} + 4(x_{5j} - x_{6j} - l_{gap})) x_{1j} x_{3j} + ((2x_{7j} + 4(x_{5j} - x_{6j})) (2x_{2j} x_{4j})) d_{copper} \leq 2.0 \text{ kg}$$

(5) Efficiency (E) &gt; 0.15

$$\frac{1}{115x_{7j}} \left( 113x_{7j} - \left( \frac{\mu_r x_{1j} (2x_{7j} + 4(x_{5j} - x_{6j} - l_{gap}))}{x_{3j}} + \frac{2\mu_r x_{2j} (2x_{7j} + 4(x_{5j} - x_{6j}))}{x_{3j}} \right) x_{7j}^2 \right) \geq 0.15$$

(6) Torque requirement for individual motors

$$\frac{(x_{1j} x_{2j} x_{7j}^2)}{\pi \left( \frac{0.000175}{x_{7j} \mu_0 (x_{5j} - x_{6j} - l_{gap})} + \frac{0.000175}{x_{7j} \mu_0 r} + \frac{\pi(2x_{5j} + x_{6j})}{4\mu_0 \mu_r \mu_0 \mu_r} \right)} = t \in T$$

$$T = \{0.05, 0.10, 0.125, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.50\}$$

(7) Platform commonality constraints:  $(x_{ij} - x_{i,j+1})y_i = 0 \quad \forall i \in I \text{ \& } \forall j \in J, j \neq m$ (8) Integerizing constraints:  $y_i^2 - y_i = 0$ (9) Deviation of efficiency from target (70 %):  $\frac{E_j}{0.7} + d_{Eff,j}^- + d_{Eff,j}^+ = 1.0 \quad \forall j \in J$ Deviation of actual mass from target mass (0.5 kg):  $\frac{M_j}{0.5} + d_{Mass,j}^- + d_{Mass,j}^+ = 1.0 \quad \forall j \in J$ (10) Deviation variables:  $d_{Eff,j}^-, d_{Eff,j}^+ \geq 0, d_{Mass,j}^-, d_{Mass,j}^+ \geq 0$ 

The formulation (Table 15.5) uses a goal programming model to address the multi-objective (target efficiency, target mass) nature of the problem. The deviation of the actual efficiency and mass of the motor is captured using deviation variables  $d_1^{+/-}$  and  $d_2^{+/-}$ , respectively. In the objective function the undesirable negative deviation of efficiency and positive deviation of mass ( $d_1^-$  and  $d_2^+$ ) are minimized.

Here  $x_1, x_2, \dots, x_8$  correspond to the design parameters, which are number of turns in the armature, number of turns in the field, area of the armature, area of the

**Table 15.6** Results from the single platform formulation

	Motor									
	1	2	3	4	5	6	7	8	9	10
X <sub>1</sub>	944.96	1,078.61	1,085.41	1,088.65	1,094.58	1,100.44	1,101.90	1,102.57	1,100.56	1,123.78
X <sub>2</sub>	70	70	70	70	70	70	70	70	70	70
X <sub>3</sub>	0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.28
X <sub>4</sub>	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34
X <sub>5</sub>	1.72	2.15	2.40	2.62	2.99	3.25	3.36	3.41	3.46	3.32
X <sub>6</sub>	5.91	5.91	5.91	5.91	5.91	5.91	5.91	5.91	5.91	5.91
X <sub>7</sub>	3.04	3.28	3.45	3.60	3.97	4.33	4.71	5.08	5.37	6.00
X <sub>8</sub>	1.62	1.62	1.62	1.62	1.62	1.62	1.62	1.62	1.62	1.62
Efficiency	0.81	0.72	0.70	0.69	0.66	0.63	0.58	0.55	0.49	0.43
Mass	0.35	0.51	0.54	0.57	0.59	0.66	0.70	0.74	0.76	0.77

**Table 15.7** Results from individual optimization

	Motor									
	1	2	3	4	5	6	7	8	9	10
X <sub>1</sub>	1,019	1,020	1,021	1,021	1,029	1,011	1,024	1,021	1,020	1,022
X <sub>2</sub>	57	65	69	75	66	57.4	61	54	58	54
X <sub>3</sub>	0.256	0.215	0.214	0.225	0.218	0.201	0.229	0.218	0.239	0.248
X <sub>4</sub>	0.272	0.258	0.255	0.251	0.217	0.201	0.232	0.238	0.234	0.243
X <sub>5</sub>	2.06	2.24	2.24	2.22	2.16	5.49	2.23	2.29	2.37	2.49
X <sub>6</sub>	5.94	5.72	5.71	5.69	5.56	4.84	5.43	5.55	5.56	5.52
X <sub>7</sub>	3.19	3.62	3.72	3.73	4.1	2.38	5.62	5.36	5.13	5.82
X <sub>8</sub>	1.2	1.47	1.65	1.84	2.32	2.3	2.5	2.8	3.12	3.1
Mag intensity	3,543	3,160	4,817	4,981	5,000	5,000	5,000	5,000	5,000	5,000
Efficiency	0.817	0.72	0.705	0.7	0.635	0.59	0.564	0.548	0.508	0.454
Mass	0.33	0.39	0.415	0.45	0.5	0.56	0.63	0.694	0.733	0.78

field wire, radius of the motor, thickness of the stator, current drawn by the motor, and stack length. The individual optima correspond to the best performance that can be achieved for each universal motor in the family. The performances of individually optimized motors serve as benchmarks to determine the loss of performance for the family members supported by the identified common platform.

The benchmark for efficiencies and mass obtained after individual optimization for the product instances are 81.7, 72, 70.5, 70, 63.5, 59.0, 56.4, 54.8, 50.8, and 45.4 % and 0.33, 0.39, 0.415, 0.45, 0.5, 0.56, 0.63, 0.694, 0.733, and 0.78 kg, respectively. The magnetizing intensity for all motors is within the allowable limit of 5,000 A turns/m. After establishing the benchmarks the number of platforms counter “k” is initiated.

The function used to evaluate the performance of products leveraged from the platform is

$$\Delta_j = \pm \left( N_j^1 \times \text{Efficiency}_j^* - N_j^1 \times \text{Efficiency}_j \right) \pm \left( N_j^2 \times \text{Mass}_j^* - N_j^2 \times \text{Mass}_j \right) \tag{15.6}$$

Here  $N_j^1$  and  $N_j^2$  are the corresponding scaling factors that can be used to scale the corresponding benchmark performances to 1. The sign conventions introduced in Sect. 15.3.2 are used to assign positive or negative signs to the value of  $\Delta_j$  obtained from the equation.  $\text{Efficiency}_j^*$ ,  $\text{Mass}_j^*$  are the normalized benchmark values for motors, and  $\text{Efficiency}_j$ ,  $\text{Mass}_j$  are the efficiency and mass of the motors leveraged using the platform. Table 15.8 shows the evaluation of products leveraged from platform 1. The limiting  $\Delta_j$  value was decided as 0.2. Motors 1, 5, 6, 7, 8, 9, and 10 show loss of performance within acceptable limits. The motors with performance loss due to commonality higher than 0.125 (motors 2, 3, and 4) were separated out to be leveraged from a second platform.

**Table 15.8** Evaluation of products leveraged from platform

	Efficiency (normalized)				Weight (normalized)				$\Delta$ Total	Feasibility
	Weight	Bench	Family	$\Delta$ Eff	Weight	Bench	Family	$\Delta$ Mass		
Motor 1	1.2240	1.0000	0.9914	0.0086	3.0303	1.0000	1.0727	0.0727	0.0813	Y
Motor 2	1.3889	1.0000	1.0056	-0.0056	2.5641	1.0000	1.3051	0.3051	0.2996	N
Motor 3	1.4184	1.0000	0.9929	0.0071	2.4096	1.0000	1.3084	0.3084	0.3155	N
Motor 4	1.4286	1.0000	0.9814	0.0186	2.2222	1.0000	1.2622	0.2622	0.2808	N
Motor 5	1.5748	1.0000	1.0409	-0.0409	2.0000	1.0000	1.1840	0.1840	0.1431	Y
Motor 6	1.6949	1.0000	1.0610	-0.0610	1.7857	1.0000	1.1768	0.1768	0.1158	Y
Motor 7	1.7730	1.0000	1.0337	-0.0337	1.5873	1.0000	1.1111	0.1111	0.0774	Y
Motor 8	1.8248	1.0000	1.0018	-0.0018	1.4409	1.0000	1.0620	0.0620	0.0601	Y
Motor 9	1.9685	1.0000	0.9547	0.0453	1.3643	1.0000	1.0355	0.0355	0.0807	Y
Motor 10	2.2026	1.0000	0.9537	0.0463	1.2821	1.0000	0.9910	-0.0090	0.0373	Y

### 15.4.3 Platform Relaxation Stage

In this stage only the nonconforming products from the platform evaluation stage are considered. The general platform relaxation formulation presented previously is applied to motors 2, 3, and 4; the optimization formulation for the universal motor family is shown in Table 15.9. The objective function in this case consists of minimization of positive deviation in mass from the target and negative deviation of efficiency. The bounds on the design variables are same as single platform formulation.

All  $y_i$  parameters associated with the scale parameters in the previous platform are given a value of 0 to hold them as scale parameters. All the platform parameters are initiated as platform parameters and held to the value obtained from the previous platform (constraints 2 and 3). There were five platform parameters in platform 1. The relaxation formulation selects a platform parameter from these five platform parameters and converts it to scale parameters so that motors with acceptable performance can be identified.

The remaining four platform parameters will have the same value as platform 1. This is achieved by using constraint 4 (Table 15.9). The constraint can only be satisfied if four of the  $y_i$  parameters have a value of 1 and the remaining one 0. This constraint restricts the continuous  $y_i$  parameters to accept only binary values and also helps in selecting the best four platform parameters from the initial five platform parameters. All the remaining constraints are same as single platform formulation, except that they are only applied to the concerned motors 2, 3, and 4.

Table 15.10 shows the values of product parameters and product performances obtained from the platform relaxation formulation. Parameter  $X_2$  was converted from a platform parameter to a scaling parameter.

Significant improvement can be seen in efficiency of the motors but at the expense of mass of the motors (Table 15.11). Efficiency of motors 2, 3, and 4 is higher than the benchmark motors. Since efficiency is a higher target (positive valued in this case), a negative sign is assigned, and since the mass of motors are higher than benchmark, which is undesirable, a positive sign is assigned to combine the values. The combined value,  $\Delta\text{Total}$ , for the three motors is 0.0581, 0.0941, and 0.1105, which is less than the allowed value of 0.2. Hence, further cascading is not necessary. Table 15.12 shows the combined parameter values and performance of motors derived from platforms 1 and 2.

The platform leveraging strategy for the universal electric motor family is shown in Fig. 15.3, which relates the platform from which each product family member is leveraged and the configuration of each platform in terms of platform parameters and scale parameters.

**Table 15.9** Platform relaxation formulation applied to universal electric motor family

$\text{Minimize } \sum_{j=2,3,4} f(d_{Mass,j}^+) + \sum_{j=2,3,4} f(d_{Eff,j}^-)$		
<p><i>Subject to</i></p>		
(1) Bounds on the design variables in	$100 \leq x_{1j} \leq 1,500$ turns $1 \leq x_{5j} \leq 1$ cm $1 \leq x_{2j} \leq 500$ turns $0.5 \leq x_{6j} \leq 10$ mm $0.01 \leq x_{3j} \leq 1$ mm <sup>2</sup> $1 \leq x_{7j} \leq 6$ amps $0.01 \leq x_{4j} \leq 1$ mm <sup>2</sup> $0 \leq x_{8j} \leq 10$ cm	
(2) Platform commonality decision variables		
	$y_1, y_5, y_7 = 0, 0 \leq y_2, y_3, y_4, y_6, y_8 \leq 1$	
(3) Platform commonality constraints (Cascading)	$(x_{2j} - 70)y_2 = 0$ $(x_{6j} - 5.91)y_5 = 0$ $(x_{3j} - 0.28)y_3 = 0$ $(x_{8j} - 1.62)y_8 = 0$ $(x_{4j} - 0.34)y_4 = 0$	
(4) Cascading constraints		
	$\sum y_i^3 \geq 4$ and $\sum y_i \leq 4; i = 2, 3, 4, 6, 8$	
(5) Magnetic intensity of each motor less than 5,000 A turns/m	$\frac{\pi(2x_{5j} + x_{6j})/2 + (2(2x_{5j} - x_{6j}) - 0.007) + 2(4.007)}{(2x_{2j}x_{7j})} \leq 5000$	
(6) Feasible geometry: $x_{6j} < x_{5j}; j = 2, 3, 4$		
(7) Mass of motor (M) < 2.0 kg	$\pi d_{steel} x_{7j} (x_{2j}^2 - (x_{5j} - x_{6j})^2) + \pi d_{steel} x_{7j} (x_{2j}^2 - (x_{5j} - x_{6j} - l_{gap})^2) (2x_{7j} + 4(x_{5j} - x_{6j} - l_{gap})) x_{1j} x_{3j} + ((2x_{7j} + 4(x_{5j} - x_{6j})) (2x_{2j} x_{4j})) d_{copper} \leq 2.0 \text{ kg}$	
(8) Efficiency (E) > 0.15	$\frac{1}{113x_{7j}} \left( 113x_{7j} \left( \frac{\mu_r x_{1j} (2x_{7j} + 4(x_{5j} - x_{6j} - l_{gap}))}{x_{5j}} + \frac{2\mu_r x_{2j} (2x_{7j} + 4(x_{5j} - x_{6j}))}{x_{3j}} \right) x_{7j}^2 \right) \geq 0.15$	
(9) Torque requirement for individual motors	$\frac{\pi \left( \frac{0.000175}{x_{7j} \mu_0 (x_{5j} - x_{6j} - l_{gap})} + \frac{0.000125}{x_{7j} \mu_0 \mu_r} + \frac{\pi (2x_{5j} + x_{6j})}{4\mu_0 \mu_r x_{6j} x_{7j}} \right)}{(x_{1j} x_{2j} x_{7j}^2)} = \{0.10, 0.125, 0.15, 0.20\}$	
(10) Deviation of actual efficiency from target efficiency (70 %)	$\frac{E_j}{0.7} + d_{Eff,j}^- - d_{Eff,j}^+ = 1.0; j = 2, 3, 4$	
(11) Deviation of actual mass from target mass (0.5 kg)	$\frac{M_j}{0.5} + d_{Mass,j}^- - d_{Mass,j}^+ = 1.0$	

**Table 15.10** Optimum design variables and performances obtained from the relaxation formulations

	Motor									
	1	2	3	4	5	6	7	8	9	10
X <sub>1</sub>	–	1,018.00	1,021.00	1,500.00	–	–	–	1,102.57	–	–
X <sub>2</sub>	–	78	86	69	–	–	–	70	–	–
X <sub>3</sub>	–	0.28	0.28	0.28	–	–	–	0.28	–	–
X <sub>4</sub>	–	0.34	0.34	0.34	–	–	–	0.34	–	–
X <sub>5</sub>	–	2.15	2.29	2.00	–	–	–	3.41	–	–
X <sub>6</sub>	–	5.91	5.91	5.91	–	–	–	5.91	–	–
X <sub>7</sub>	–	3.27	3.32	3.64	–	–	–	5.08	–	–
X <sub>8</sub>	–	1.62	1.62	1.62	–	–	–	1.62	–	–
Eff.	–	0.80	0.78	0.72	–	–	–	0.55	–	–
Mass	–	0.46	0.50	0.57	–	–	–	0.74	–	–

## 15.5 Conclusions

A single platform, in most cases, is insufficient to design a family of products while using platform approach. Single platform approach assumes that when a component or a product parameter is shared, it is shared across all products in the family. As the number of products in the family increases or the portfolio of different products varies considerably, a single platform approach may lead to product family members with inferior performance. In a multi-platform approach, the family members are leveraged from more than one platform so that products with minimal loss of performance can be achieved. Cost efficiency of single platform design may be higher compared to multi-platform design as an increase in number of platforms will lead to increase in cost of the derived product family. In a multi-platform design it is therefore necessary to design the family of products using optimum number of platforms. Also in case of multi-platform design, the combination of products that are leveraged from each platform and the configuration of each platform leading to a family of products with minimal loss of performance need to be determined.

The platform relaxation method is a multi-platform optimization method for the design of scalable product families. The inputs to the formulation are (1) the specification of the product family members, (2) the underlying mathematical model that related the product parameters to performances, and (3) the identification of platform parameters (optional). The platform relaxation method takes a holistic view of the entire product family design process. The mathematical model developed for single platform design is capable of representing both the product platform and the product variants. During the single platform stage of the design process, both the platform and the product variants are simultaneously optimized. Trade-off is performed between the number of platform parameters and the loss of performance due to commonality to arrive at the optimum platform and the optimum product instances. The platform relaxation method converts the binary

**Table 15.11** Evaluation of products leveraged from platform 2

	Efficiency (normalized)			Weight (normalized)			$\Delta$ Total	Feasibility	
	Weight	Bench	Family	$\Delta$ Eff	Weight	Bench			Family
Motor 2	1.3889	1.0000	1.1111	-0.1111	2.5641	1.0000	1.1692	0.1692	Y
Motor 3	1.4184	1.0000	1.1064	-0.1064	2.4096	1.0000	1.2048	0.2048	Y
Motor 4	1.4286	1.0000	1.0229	-0.0229	2.2222	1.0000	1.1333	0.1333	Y



**Table 15.12** Combined results for Platforms 1 and 2

	Motor									
	1	2	3	4	5	6	7	8	9	10
X <sub>1</sub>	944.96	1,018.00	1,021.00	1,500.00	1,094.58	1,100.44	1,101.90	1,102.57	1,100.56	1,123.78
X <sub>2</sub>	70	78	86	69	70	70	70	70	70	70
X <sub>3</sub>	0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.28
X <sub>4</sub>	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34
X <sub>5</sub>	1.72	2.15	2.29	2.00	2.99	3.25	3.36	3.41	3.46	3.32
X <sub>6</sub>	5.91	5.91	5.91	5.91	5.91	5.91	5.91	5.91	5.91	5.91
X <sub>7</sub>	3.04	3.27	3.32	3.64	3.97	4.33	4.71	5.08	5.37	6.00
X <sub>8</sub>	1.62	1.62	1.62	1.62	1.62	1.62	1.62	1.62	1.62	1.62
Eff.	0.81	0.80	0.78	0.72	0.66	0.63	0.58	0.55	0.49	0.43
Mass	0.35	0.46	0.50	0.51	0.59	0.66	0.70	0.74	0.76	0.77

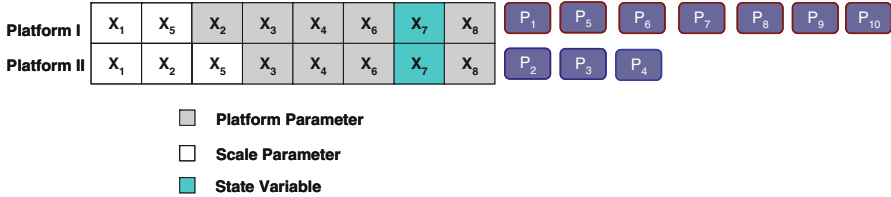


Fig. 15.3 Platform relaxation strategy for universal electric motor case study

platform commonality parameters to continuous parameters to enable the formulation to be implemented in a gradient-based optimization method. The model is constrained mathematically to accept only binary values in the end for the platform commonality parameters. The formulation developed is easy to implement in gradient-based optimization methods.

Relaxation of platform parameters is used to leverage the family when multiple platforms are required. During relaxation, one of the platform parameters is relaxed to a scale parameter so that products with lesser loss of performance can be identified. This reduces the number of platform parameters from the previous platform, which in turn can lead to products with better performances. In the platform relaxation method, the modeling approach is similar to that of single platform formulation. The platform, product instances, and platform commonality are modeled in the relaxation formulation. Both single platform and relaxation formulations initially convert the MINLP to a continuous problem and then constrain the solution to discrete spaces. In case of the relaxation formulation, constraints are simultaneously used to select the platform parameters and also to constraint the model to accept only binary values for commonality parameters.

In the platform relaxation method, the number of platforms required to support the platform is not modeled as part of the formulation. Instead the initial platform is relaxed until all the products with acceptable loss of performance are leveraged. The number of platforms required to support the family depends on the threshold value of the acceptable loss of performance, due to commonality and the path chosen by the designer after the evaluation of products. Since the platform relaxation method uses the single platform and cascades it to generate the subsequent platforms, the approach maintains a relation between the product platforms which can lead to commonality within the different platforms. An evaluation function is used to determine the loss of performance of the product family members due to commonality. If the loss of performance due to commonality for any of the products in the family is greater than a user specified value, a multi-platform approach is used.

Platform relaxation method does not require the identification of platform parameters by the designer; however, it gives the designer the flexibility of being able to specify the platform. The outputs from the method are (1) the different product platforms and the products that are supported by the platforms and (2) the product family instances and their performances. Other secondary information like

loss of performance due to commonality in comparison to benchmarks and the best possible performance of products can be obtained from the formulation. As evident from earlier discussions, the method is comprised of different stages. The method is only applicable towards scalable product families, wherein each product instance in the family can be completely described by the same set of product parameters. Hence, the method will fall under the category of multistage, non-platform-specified, scale-based product family design method.

## References

- Belloni A, Freund R, Selove M, Simester D (2008) Optimizing product line designs: efficient methods and comparisons. *Manag Sci* 54(9):1544–1552
- Chapman SJ (1991) *Electric machinery fundamentals*. McGraw-Hill, New York, NY
- Dai Z, Scott M (2007) Product platform design through sensitivity analysis and cluster analysis. *J Intell Manuf* 18(1):97–113
- Fellini R, Kokkolaras M, Papalambros PY (2006) Quantitative platform selection in optimal design of product families, with application to automotive engine design. *J Eng Des* 17(5):429–446
- Gao F, Xiao G, Simpson T (2009) Module-scale-based product platform planning. *Res Eng Des* 20(2):129–141
- Khajavirad A, Michalek JJ (2007) An extension of the commonality index for product family optimization. In: *Proceedings of the ASME IDETC/CIE conference 2007, DETC2007-35605*
- Khire R, Messac A, Simpson TW (2006) Optimal design of product families using selection-integrated optimization (SIO) methodology. In: *11th AIAA/ISSMO symposium on multidisciplinary analysis and optimization*, 6–8 Sept 2006, Portsmouth, VA, AIAA-2006-6924
- Messac A (1996) Physical programming: effective optimization for computational design. *AIAA J* 34(1):149–158
- Messac A, Martinez MP, Simpson TW (2002) A penalty function for product family design using physical programming. *ASME J Mech Des* 124(2):164–172
- Moon SK, Park K, Simpson TW (2011) Platform strategy for product family design using particle swarm optimization. In: *Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference IDETC/CIE*, Washington, DC, USA, 28–31 Aug 2011. DETC2011-48060
- Nayak RU, Chen W, Simpson TW (2002) A variation-based method for product family design. *Eng Optim* 34(1):65–81
- Nelson S, Parkinson M, Papalambros P (2001) Multi criteria optimization in product family design. *J Mech Des* 123:199–204
- Simpson TW (1998) A concept exploration method for product family design. Ph.D. Dissertation, G.W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA
- Simpson T (2006) Methods for optimizing product platforms and product families. In: Simpson T, Siddique Z, Jiao JR (eds) *Product platform and product family design*. Springer, New York, NY, pp 133–156
- Simpson TW, D'Souza B (2004) Assessing variable levels of platform commonality within a product family using a multi-objective genetic algorithm. *Concur Eng: Res Appl* 12(2):119–130
- Simpson TW, Maier JRA, Mistree F (2001) Product platform design: method and application. *Res Eng Des* 13(1):2–22
- Veinott CG, Martin JE (1986) *Fractional and subfractional horsepower electric motors*. McGraw-Hill, New York, NY

**PART III**  
**Product Family Development**  
**and Implementation**

# Chapter 16

## Global Product Family Design: Simultaneous Optimal Design of Module Commonalization and Supply Chain Configuration

Kikuo Fujita

**Abstract** Global product family design is the problem in which product variants and supply chain configuration are simultaneously designed. It has become a significant concern of manufacturing industries under globalization. In this chapter, simultaneous design of module commonalization and supply chain configuration is formulated as a multi-objective mixed-integer programming problem under the criteria on quality, cost, and delivery. Then, an optimization algorithm for obtaining Pareto optimal solutions is configured by using a neighborhood cultivation genetic algorithm and simplex method, and a clustering technique of such Pareto solutions is introduced with a principal component analysis method for investigating the optimality and compromise in global product family design. Finally, some numerical case studies are demonstrated.

### 16.1 Introduction

Product family and platform design have become an essentially important concept for meeting with the demands on mass customization (Pine 1993) and enhancing the product performance (Simpson et al. 2005; Jiao et al. 2007). On the other hand, as design and manufacturing have become worldwide under the globalization, not only the markets but also allocations of various levels of factories are spread over

---

An earlier version of this chapter appeared in K. Fujita, K. Nasu, Y. Ito, and Y. Nomaguchi, (2012) Global Product Family Design: Multi-Objective Optimization and Design Concept Exploration, ASME Design Engineering Technical Conferences - Design Automation Conference, Chicgao, IL, ASME, Paper No. DETC2012/DAC-70858 (© ASME 2012), reprinted with permission.

K. Fujita (✉)

Department of Mechanical Engineering, Graduate School of Engineering,  
Osaka University, 2-1, Yamadaoka, Suita, Osaka 565-0871, Japan  
e-mail: [fujita@mech.eng.osaka-u.ac.jp](mailto:fujita@mech.eng.osaka-u.ac.jp)

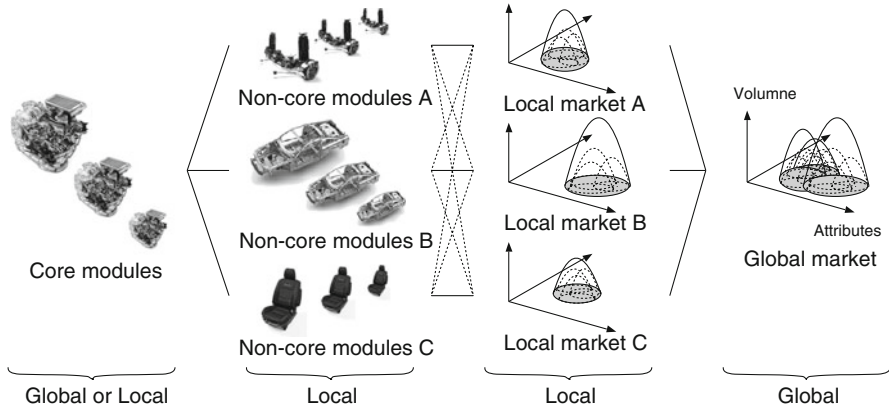
the globe. The simultaneous design of product variants and supply chain configuration (Simchi-Levi et al. 1999) has become a new and relevant standpoint toward the next level of the optimality in product design and development. Such a design situation is called “global product family design” (Fujita et al. 2012a, b). The context of global product family is not only complicated under various factors and their interactions but also vague in strategic decision making. It is difficult to identify the optimal design only with numerate supports. Toward any literate supports, the framework of design concept exploration is something promising (Obayashi and Sasaki 2003; Fujita et al. 2011). In design concept exploration, after a large set of Pareto optimal solutions are generated typically with multi-objective genetic algorithms (Deb 2001), they are categorized into a small set of clusters for identifying their conceptual meanings.

This chapter discusses multi-objective optimization and design concept exploration for enhancing global product family design. In the following sections, Sect. 16.2 reviews their backgrounds. Section 16.3 develops a multi-objective mixed-integer formulation for the simultaneous design of module commonalization and supply chain configuration. Section 16.4 configures an optimization algorithm for obtaining Pareto optimal solutions by using a neighborhood cultivation genetic algorithm. It also reports an application of the principal component analysis method for analyzing the tendencies of Pareto solutions. Then, Sect. 16.5 demonstrates some numerical case studies for ascertaining the validity and promise of the proposed mathematical model and optimal design techniques toward the excellence in global product family design.

## 16.2 Global Product Family and Its Design Problem

### 16.2.1 *Product Family Under Global Manufacturing*

The concept of product family and platform brings various possibilities for meeting with various voices of customers through the flexibility under modular architecture (Ulrich 1995). Such voices are diverse even within a market, and their tendency is different among various markets due to the phase of economical growth, lifestyle, culture, etc. On the other hand, a product consists of a core module or component, which dominates its key features, and other subsidiary modules and components, which are necessary for realizing the whole functionality of a product. Among them, the production of the former may be restricted geographically, but one of the latter may not be restricted. Further, locations for producing modules or components, ones for assembling them into products, and ones where they are consumed can be unique or rather different. The best combination of these alternatives should be determined by compromising various trade-offs. This poses the design problem of what we call “global product family design” (Fujita et al. 2012a, b).



**Fig. 16.1** Decisions on globality versus locality in manufacturing

When viewing the above standpoints on global product family as a whole, several decisions on globality or locality are required. Its scenario can be illustrated as shown in Fig. 16.1. That is, whether the core modules are produced at the key factory or at distributed factories must be determined as shown in the left-hand side of the figure. Following such a decision, the contents of core module may be affected by the available production technology, possible scales of production resources, etc. On the other hand, noncore modules could be produced and assembled into products at any distributed factories if the economy of scale is relevant, for instance, when a local market has sufficient demands and so forth, as shown in the middle of the figure. However, various resources required for design and manufacturing activities are linked with each other among local activities and through global operation. The global market must be explicitly taken into consideration for achieving the global success, as shown in the right-hand side of the figure.

### 16.2.2 Multi-objectiveness and Concept-Level Design

Since global product family relates to board issues, the optimality measure would be spread over various criteria, such as cost, quality, and delivery, at least. Further, the decision must be done in the especially early phase of design and development. The former point is similar to the situation of multidisciplinary design optimization (MDO). This means that multi-objective optimization must be the concern. The latter point means that the concept-level strategic decision making must be considered over operational support through mathematical frameworks.

When viewing the history of optimal design paradigm, MDO has focused on more complicated design problems which include different system behaviors. Such a focus requires the techniques for efficiently obtaining a set of Pareto optimal

solutions. Multi-objective genetic algorithms (MOGA) have become typical ones for such purpose. Since a set of Pareto optimal solutions corresponds to a set of relevant alternatives under the given requirements, they are effective for supporting designer's decision making, but they still remain as a set of meaningless snatches. It is necessary to interpret the tendency that dominates over Pareto solutions, and such interpretations must correspond to what we call design concepts.

Design concepts include the mapping relationship among requirements and the contents of entities. The superior, feasible, and affordable direction of designs should be identified with considering the compromise among various factors. Thus, if the landscape of Pareto solutions is conceptually interpreted somehow, it must be useful toward the excellence of global product family. The design concept exploration is a means for such purpose (Obayashi and Sasaki 2003; Fujita et al. 2011). In the design concept exploration, a set of Pareto solutions generated by any optimization technique are categorized into a small set of clusters by data-mining techniques such as the self-organizing map (SOM), the principal component analysis (PCA). Those paradigms have been successfully applied to multidisciplinary optimization of aircraft design (Obayashi and Sasaki 2003; Shimizu et al. 2008) and photovoltaic system design (Sato and Wakao 2010).

The following sections investigate an optimal design method for a class of global product family design problems under the above standpoint.

## 16.3 Mathematical Model of Global Product Family Design

### 16.3.1 *Integration of Product Family and Supply Chain*

This chapter focuses on a strategy-level integrated design problem of product family and supply chain as an instance of the design problems on global product family (Fujita et al. 2012a). Figure 16.2 summarizes its outline, which consists of two layers. The upper layer represents the contents of a product family (Fujita et al. 1999). That is, a product is composed of modules, and it has a fixed number of module slots, as shown in the middle part. Each variant is configured by selecting modules from their candidates, which is shown in the left part, for respective slots. They are targeted to various markets, which is shown in the right part. The lower layer represents the components of supply chain. That is, sites of module production; ones of product production, which can be assembly operations; and ones of product sales are geographically different and distant. How to connect them as a network is the strategy-level problem of supply chain configuration. It is not necessary to utilize all possible routes, and it is important to utilize cost-effective ones. Such contents must be strategically determined within various possibilities and restrictions. In the figure, module design and selection of module production sites and product design and selection of product production sites are linked among these two layers, respectively. That is, it is necessary to simultaneously solve the



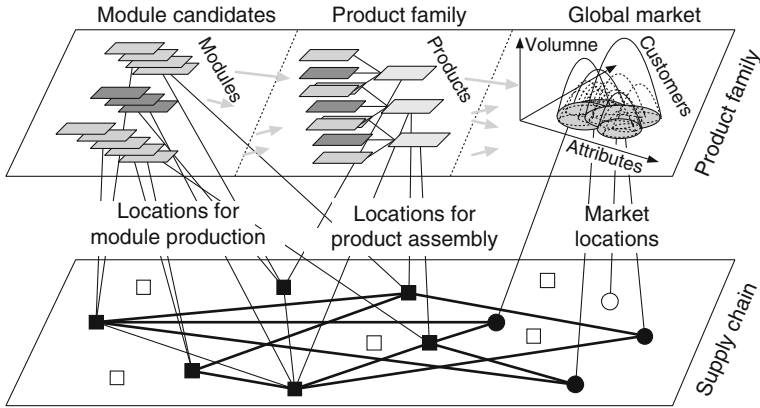


Fig. 16.2 Integrated problem of product family and supply chain

product family design problem and the supply chain configuration problem with considering realization of product contents, merits from commonalization of modules, flexibility in diversion of distribution network, etc.

### 16.3.2 Conditions for Problem Formulation

For studying the simultaneous design problem of product family design through module commonalization and supply chain configuration, this chapter introduces the following conditions and assumptions.

Regarding the contents of products as a product family, the following are introduced:

1. A unique product family is considered. That is, any interaction among product families are not considered.
2. The product is configured with a modular architecture. The contents of modules are given as conditions. That is, each product is directly composed of a set of modules, while any sub-modules or subassemblies are not considered.
3. A product  $P_i$  ( $i = 1, 2, \dots, I$ ) has a certain number of module slots  $S_j$  ( $j = 1, 2, \dots, J$ ). Each product is configured with selecting suitable modules from their candidates  $m_{j,k}$  ( $k = 1, 2, \dots, K_j$ ) for respective slots.
4. Arbitrary combination of modules is evenly accepted within the given constraints.

Regarding manufacturing activities, the followings are introduced:

5. Each factory has a capability for producing various modules and products simultaneously within its capacity.
6. The resource limitation on the company's actions is substituted with the constraints of module and product production volumes.

7. Cost for module and product production is represented with the sum of fixed cost and variable cost. Learning effects of production are not considered.

Regarding supply chain characteristics, various sites are denoted with  $M^l$  ( $l = 1, 2, \dots, L$ ). The followings are introduced:

8. The supply chain structure, i.e., sites for module production, product production (assembly), market sales,<sup>1</sup> and transportation among them are considered. Any stock at any warehouses are not considered.

9. Transportation capability for each route is also restricted with the sum of modules and products.

10. Cost for transportation is represented with variable cost. Any trade customs and money exchanges are not considered.

Regarding the market and business circumstances, the followings are introduced:

11. As the diversity of markets, a set of local markets are considered. They are geographically distant, and their demand distribution patterns are different.

12. Prices of products are fixed according to the locations of respective markets.

13. The demand volumes are given, and all supplied units are sold.

14. Cost for sales is represented with fixed cost. The contents of fixed cost are represented with the sum of one depended on sites and one depended on module or product kinds.

15. The competition with other companies is not considered.

16. Temporal factors are not considered. For example, phenomena within a single financial period are considered.

Further, for building a mathematical model on the above design problem, the following assumptions are introduced:

17. The contents of decision are whether modules are produced or not, whether products are produced or not, whether products are sold or not, and how many they are if they are produced or sold.

18. Each disjunctive decision such as whether production or sales is performed at a site is modeled with a 0–1 integer variable.

19. While each quantitative decision, such as production volume and transportation volume, is primality represented with zero or positive integer, it is substituted with nonnegative real number for reducing the scale of mathematical problem.

Under these conditions, the mathematical model becomes to be constructed with 0–1 design variables on the choice of modules and various sites and nonnegative continuous design variables on the volumes of production and transportation, and the criteria are defined on cost, quality, and delivery, respectively. The following two subsections explain its details, which is developed through extending the single-objective one (Fujita et al. 2012a) to multi-objective one (Fujita et al. 2012b).

---

<sup>1</sup>For convenience, the supersuffixes,  $s \in \text{Module}$ ,  $t \in \text{Product}$ ,  $r \in \text{Market}$ , are used on the symbols on sites in the followings.

### 16.3.3 Product Family Model

#### 16.3.3.1 Module Production

Whether Module  $m_{j,k}$ , which is the  $k$  th module candidate to be embedded into the  $j$  th module slot  $S_j$  of products, is produced at Site  $s$  ( $s \in Module$ ) or not is determined with the following 0–1 variable:

$$\delta_{j,k}^s = \begin{cases} 1 \cdots \text{Module } m_{j,k} \text{ is produced at Site } s \\ 0 \cdots \text{Module } m_{j,k} \text{ is not produced at Site } s \end{cases} \quad (16.1)$$

On the other hand, whether Module  $m_{j,k}$  can be produced at Site  $s$  is represented with the following coefficient:

$$\alpha_{j,k}^s = \begin{cases} 1 \cdots \text{Module } m_{j,k} \text{ is produced at Site } s \\ 0 \cdots \text{Module } m_{j,k} \text{ is not produced at Site } s \end{cases} \quad (16.2)$$

Since the capability of module production depends on sites, the following constraint is assigned on the above variables:

$$(1 - \alpha_{j,k}^s) \delta_{j,k}^s = 0 \quad (16.3)$$

The production volume of Module  $m_{j,k}$  at Site  $s$  is denoted as  $x_{j,k}^{s \rightarrow}$ . Then, production volume  $x_{j,k}^{s \rightarrow}$  of Module  $m_{j,k}$  must satisfy the following constraint at least:

$$W_{MB}^s \delta_{j,k}^s \leq w_{j,k}^s x_{j,k}^{s \rightarrow} \leq W_M^s \alpha_{j,k}^s \delta_{j,k}^s \quad (16.4)$$

where the unit of this constraint is, for example, the volume of consumed materials or the number of works, which is representative for the production capability of a factory.  $w_{j,k}^s$  is the coefficient,  $W_{MB}^s$  is the constant representing its lower bound, and  $W_M^s$  is the constant representing its upper bound.

#### 16.3.3.2 Module Transportation

The transportation volume of Module  $m_{j,k}$  from Site  $s$  ( $s \in Module$ ) to Site  $t$  ( $t \in Product$ ) is denoted as  $x_{j,k}^{s \rightarrow t}$ . The production volume  $x_{j,k}^{s \rightarrow}$  of Module  $m_{j,k}$  at Site  $s$  must satisfy the following constraint:

$$x_{j,k}^{s \rightarrow} = \sum_t x_{j,k}^{s \rightarrow t} \quad (16.5)$$

The total module transportation volume from Site  $s$  to Site  $t$  must satisfy the following constraint:

$$\sum_j \sum_k w_{j,k}^{s \rightarrow t} x_{j,k}^{s \rightarrow t} \leq W_M^{s \rightarrow t} \quad (16.6)$$

where the unit of this constraint is, for example, the total weight or the total cubic capacity, which is representative for the transportation capability of this route.  $w_{j,k}^{s \rightarrow t}$  is the coefficient, and  $W_M^{s \rightarrow t}$  is the constant representing its upper bound.

### 16.3.3.3 Product Production

Whether Product  $P_i$  is produced at Site  $t$  ( $t \in Product$ ) or not is determined with the following 0–1 variable:

$$\delta_i^{t \rightarrow} = \begin{cases} 1 \cdots \text{Product } P_i \text{ is produced at Site } t \\ 0 \cdots \text{Product } P_i \text{ is not produced at Site } t \end{cases} \quad (16.7)$$

On the other hand, whether Product  $P_i$  is producible at Site  $t$  or not is represented with the following coefficient:

$$\beta_i^t = \begin{cases} 1 \cdots \text{Product } P_i \text{ is produced at Site } t \\ 0 \cdots \text{Product } P_i \text{ is not produced at Site } t \end{cases} \quad (16.8)$$

Since the capability of product production depends on sites, the following constraint is assigned on the above variables:

$$(1 - \beta_i^t) \delta_i^{t \rightarrow} = 0 \quad (16.9)$$

The supply volume  $x_{j,k}^{\rightarrow t}$  of Module  $m_{j,k}$  at Site  $t$  is represented with the following equation by using module transportation volumes  $x_{j,k}^{s \rightarrow t}$  ( $s \in Module$ ):

$$x_{j,k}^{\rightarrow t} = \sum_s x_{j,k}^{s \rightarrow t} \quad (16.10)$$

If any product is not produced at Site  $t$ , it is not necessary to supply any modules to Site  $t$ . Thus, the following constraint is imposed:

$$\sum_j \sum_k x_{j,k}^{\rightarrow t} = 0 \quad \text{if} \quad \sum_i \delta_i^{t \rightarrow} = 0 \quad (16.11)$$

Whether Module  $m_{j,k}$  is available for implementing to Product  $P_i$  or not is represented with the following 0–1 integer coefficient (Fujita et al. 1999):

$$\gamma_{j,k \rightarrow i} = \begin{cases} 1 \cdots \text{Module } m_{j,k} \text{ is available for Product } P_i \\ 0 \cdots \text{Module } m_{j,k} \text{ is not available for Product } P_i \end{cases} \quad (16.12)$$

The assignment volume of Module  $m_{j,k}$  to Product  $P_i$  is denoted as  $x_{j,k \rightarrow i}^t$ . It must satisfy the following constraint and relationship. That is, it must be greater than or equal to the minimal volume  $W_{PB}^s$  and be less than or equal to  $x_{j,k}^{-t}$  if producing Product  $P_i$ :

$$W_{PB}^s \delta_i^{t \rightarrow} \leq x_{j,k \rightarrow i}^t \leq x_{j,k}^{-t} \gamma_{j,k \rightarrow i} \beta_i^t \delta_i^{t \rightarrow} \quad (16.13)$$

Under the above conditions, regarding the product production, the production volume  $x_i^{t \rightarrow}$  of Product  $P_i$  must satisfy the following relationships on all module slots  $S_j$ :

$$\forall j, \quad x_i^{t \rightarrow} = \sum_k x_{j,k \rightarrow i}^t \quad (16.14)$$

Regarding the total production volume at Site  $s$ , the sum of module production volume  $x_{j,k}^{s \rightarrow}$  and product production volume  $x_i^{s \rightarrow}$  must satisfy the following constraint:

$$\sum_j \sum_k w_{j,k}^s x_{j,k}^{s \rightarrow} + \sum_i w_i^s x_i^{s \rightarrow} \leq W^s \quad (16.15)$$

where  $w_{j,k}^s$  and  $w_i^s$  are the coefficients, and  $W^s$  is the constant representing its upper bound.  $w_{j,k}^s$  is identical to one used in Eq. (16.4).

#### 16.3.3.4 Product Transportation

The transportation volume of Product  $P_i$  from Site  $t$  ( $t \in \text{Product}$ ) to Site  $r$  ( $r \in \text{Market}$ ) is denoted as  $x_i^{t \rightarrow r}$ . This variable satisfies the following equation under the relationship with the product production volume:

$$x_i^{t \rightarrow} = \sum_r x_i^{t \rightarrow r} \quad (16.16)$$

The total product transportation volume from Site  $t$  to Site  $r$  must satisfy the following constraint:

$$\sum_i w_i^{t \rightarrow r} x_i^{t \rightarrow r} \leq W_P^{t \rightarrow r} \quad (16.17)$$

where the unit of this constraint is, for example, the total weight or total cubic capacity, which is representative for the transportation capability of this route.  $w_i^{t \rightarrow r}$  is the coefficient and  $W_p^{t \rightarrow r}$  is the constant representing its upper bound.

### 16.3.3.5 Product Sales

Whether Product  $P_i$  is sold at Site  $r$  ( $r \in Market$ ) or not is determined with the following 0–1 variable:

$$\delta_i^{\rightarrow r} = \begin{cases} 1 \cdots \text{Product } P_i \text{ is sold at Site } r \\ 0 \cdots \text{Product } P_i \text{ is not sold at Site } r \end{cases} \quad (16.18)$$

If Product  $P_i$  is not sold at Site  $r$ , its supply to the corresponding market is not necessary. Thus, the following constraint is assigned:

$$(1 - \delta_i^{\rightarrow r})x_i^{\rightarrow r} = 0 \quad (16.19)$$

where  $x_i^{\rightarrow r}$  is the supply volume of Product  $P_i$  to Site  $r$  is represented with the following equation by using product transportation volumes  $x_i^{t \rightarrow r}$  from Site  $t$  ( $t \in Product$ ):

$$x_i^{\rightarrow r} = \sum_t x_i^{t \rightarrow r} \quad (16.20)$$

On the other hand, the demand and supply relationship on Product  $P_i$  at Site  $r$  is represented as follows:

$$x_i^{\rightarrow r} = D_i^r \delta_i^{\rightarrow r} \quad (16.21)$$

where  $D_i^r$  is the potential demand volume of Product  $P_i$  at Site  $r$ . This equation is developed under the assumption that if a product is provided to a market, the supply volume satisfies the expected demand volume. This assumption reflects the strategic decision making on whether a specific product is supplied to a certain market or not.

The total sales through the whole lineup are calculated with the following equation by using the price  $p_i^r$  of Product  $P_i$  at Site  $r$ :

$$Sales = \sum_i \sum_r p_i^r x_i^{\rightarrow r} \quad (16.22)$$

### 16.3.4 Cost Model

#### 16.3.4.1 Fixed Costs

As fixed costs, one for module production, one for product production, one for product sales, one for facility at site, and one for equipments depending to the kinds of modules and products are estimated.

Equipment cost for respective modules is considered as the fixed cost for module production  $C_{MP}$ . It is represented with the following equation:

$$C_{MP} = \sum_j \sum_k \sum_s k_{MP}^s C_{MPj,k} \delta_{j,k}^s \quad (16.23)$$

where  $C_{MPj,k}$  is the coefficient depending on the kinds of modules.  $k_{MP}^s$  is an adjustment factor of module production facility cost by each site.

Equipment cost for respective products is considered as the fixed cost for product production  $C_{PA}$ . It is represented with the following equation:

$$C_{PA} = \sum_i \sum_t k_{PA}^t C_{PAi} \delta_i^{s \rightarrow t} \quad (16.24)$$

where  $C_{PAi}$  is the coefficient depending on the kinds of products.  $k_{PA}^t$  is an adjustment factor of product production unit cost by each site.

The sum of fundamental equipment cost  $C_S$  for module and product production per site over all sites is represented as follows with the above two kinds of  $\delta$  variables:

$$C_S = \sum_s C_S^s \left\{ 1 - \prod_j \prod_k (1 - \delta_{j,k}^s) \prod_i (1 - \delta_i^{s \rightarrow}) \right\} \quad (16.25)$$

where  $C_S^s$  is the coefficient that considered the difference among sites.

The fixed cost for product sales  $C_{PS}$  is represented with the following equation as the sum of facility cost per site and equipment cost per products:

$$C_{PS} = \sum_r \left\{ C_{PS}^r \left( 1 - \prod_i (1 - \delta_i^{\rightarrow r}) \right) + \sum_i k_{PS}^r C_{PSi} \delta_i^{\rightarrow r} \right\} \quad (16.26)$$

where  $C_{PS}^r$  and  $C_{PSi}$  are the coefficients depending on the locations of sites and the kinds of modules.  $k_{PS}^r$  is an adjustment factor of product sales fixed cost by each site.

### 16.3.4.2 Variable Costs

As variable costs, one on module production, one on module transportation, one on product production and one on product transportation are estimated.

The variable cost on module production  $c_{MP}$  depends on the production volume. It is represented with the following equation.

$$c_{MP} = \sum_j \sum_k \sum_s k_{mp}^s c_{MPj,k} x_{j,k}^{s \rightarrow} \quad (16.27)$$

where  $c_{MPj,k}$  is the coefficient that considers the difference among modules.  $k_{mp}^s$  is an adjustment factor of module production unit cost by each site.

The variable cost on module transportation  $c_{MT}$  depends on the transportation volume. It is represented with the following equation:

$$c_{MT} = \sum_s \sum_t k_{mt}^s c_{MT}^{s \rightarrow t} \left( \sum_j \sum_k c_{MTj,k} x_{j,k}^{s \rightarrow t} \right) \quad (16.28)$$

where  $c_{MTj,k}$  and  $c_{MT}^{s \rightarrow t}$  are the coefficients that substitute the transportation distance and that consider the difference among modules.  $k_{mt}^s$  is an adjustment factor of module transportation unit cost by each site.

The variable cost on product production  $c_{PA}$  is represented with the following equation:

$$c_{PA} = \sum_i \sum_t k_{pa}^t c_{PA \rightarrow i} x_i^{t \rightarrow} \quad (16.29)$$

where  $c_{PA \rightarrow i}$  is the coefficient that is different by products.  $k_{pa}^t$  is an adjustment factor of product production facility cost by each site.

The variable cost on product transportation  $c_{PT}$  is represented with the following equation:

$$c_{PT} = \sum_t \sum_r k_{pt}^t c_{PT}^{t \rightarrow r} \left( \sum_i c_{PTi} x_i^{t \rightarrow r} \right) \quad (16.30)$$

where  $c_{PTi}$  and  $c_{PT}^{t \rightarrow r}$  are the coefficients that substitute the transportation distance and that are different by products.  $k_{pt}^t$  is an adjustment factor of product transportation unit cost by each site.



### 16.3.4.3 Total Cost and Profit

The profit, which is calculated as the difference between sales and cost, can be taken as one of the objectives:

$$Profit = Sales - (C_{MP} + c_{MP} + c_{MT} + C_{PA} + c_{PA} + c_{PT} + C_{PS}) \quad (16.31)$$

### 16.3.5 Quality Model

The quality of a product depends on all phases of manufacturing such as material, fabrication, assembly, transportation, and delivery. While its indicator must be based on the contents of respective products, this chapter introduces an abstracted indicator, which is called here “quality measure,” and the range of that is  $[0 : 1]$ . First, when Module  $m_{j,k}$  is produced at Site  $s$  ( $s \in Module$ ), its quality measure is denoted as  $q_{j,k}^s$ . When Product  $P_i$  is assembled at Site  $t$  ( $t \in Product$ ), the quality measure of assembling process is denoted as  $q_i^t$ . Under these, it is assumed that the quality measure  $Q_i^t$  of Product  $P_i$  produced at Site  $t$  is represented with the following equation:

$$Q_i^t = q_i^t \left( \prod_j q_{j,k}^s \right) \quad (16.32)$$

This means that  $Q_i^t$  is all product of the quality measures of implemented modules and assembly process. Besides, if a sort of modules or products are provided to a certain site from different sites more than two, the worst quality measure is adopted.

The worst case  $Q_{min}$  of quality measures  $Q_i^t$  of all products over all sites is calculated with the following equation:

$$Q_{min} = \min_{t,i} \{Q_i^t\} \quad (16.33)$$

### 16.3.6 Delivery Model

Delivery of a product from an order to a customer is another important criterion behind cost and quality in today’s manufacturing. In this chapter, production lead time is taken as such a measure. That is, lead time for producing Module  $m_{j,k}$  at Site  $s$  ( $s \in Module$ ) is denoted as  $l_{j,k}^s$ , lead time for transporting Module  $m_{j,k}$  from Site  $s$  to Site  $t$  ( $t \in Product$ ) is denoted as  $l_m^{s \rightarrow t}$ , lead time for producing Product  $P_i$  at Site  $t$

is denoted as  $l_i^t$ , and lead time for transporting Product  $P_i$  from Site  $t$  to Site  $u$  ( $u \in Market$ ) is denoted as  $l_p^{t \rightarrow r}$ . Then, the total lead time of Product  $P_i$  from module production to product delivery is represented with the following equation:

$$L_i^u = \max_j \left\{ l_{j,k}^s + l_m^{s \rightarrow t} \right\} + l_i^t + l_p^{t \rightarrow r} \tag{16.34}$$

This means that  $L_i^u$  is the total sum of the maximum of the sums of module production time and module transportation time, time for assembling them into a product, and its time to market. When more than two modules or products are provided to a unique site, the longest lead time is taken as  $l_{j,k}^s$  and  $l_i^t$  in the above equation.

The worst case  $L_{max}$  of the lead time of all products to be produced  $L_i^u$  is calculated with the following equation:

$$L_{max} = \max_{u,i} \{L_i^u\} \tag{16.35}$$

### 16.3.7 Mathematical Formulation

Under the mathematical model developed in the above subsections, the optimal design problem of simultaneous decision of module commonalization and supply chain configuration is formulated as follows.

The design variables are the following two types:

- 0–1 integer variables on module production, product production, and product sales, that is  $(\delta_{j,k}^s, \delta_i^{s \rightarrow t}, \delta_i^{t \rightarrow r})$
- Nonnegative real variables on module production, module transportation, product production, module assignment to product, and product transportation  $(x_{j,k}^{s \rightarrow t}, x_{j,k}^{s \rightarrow t}, x_{j,k \rightarrow i}^t, x_i^{t \rightarrow r})$

Respective design variables are restricted their domain by the following constraints:

$$\delta_{j,k}^s \in \{0, 1\}, \delta_i^{s \rightarrow t} \in \{0, 1\}, \delta_i^{t \rightarrow r} \in \{0, 1\} \tag{16.36}$$

$$x_{j,k}^{s \rightarrow t} \geq 0, x_{j,k}^{s \rightarrow t} \geq 0, x_{j,k \rightarrow i}^t \geq 0, x_i^{t \rightarrow r} \geq 0 \tag{16.37}$$

Regarding objective functions, the expected profit, calculated with Eq. (16.31), is to be maximized; the worst case of quality measure across all products, calculated with Eq. (16.33), is to be maximized; and the worst case of lead time across all products, calculated with Eq. (16.35), is to be minimized. That is, the followings are taken as the objectives:

$$\left. \begin{array}{l} \max \text{ Profit} \\ \max \text{ } Q_{min} \\ \max \text{ } L_{max} \end{array} \right\} \quad (16.38)$$

Regarding constraints, ones on module production capability, Eq. (16.3); ones on production volume of respective modules, Eq. (16.4); ones on module production volume and module transportation volume, Eq. (16.5); ones on module transportation volume among respective pairs of different sites, Eq. (16.6); ones on product production capability, Eq. (16.9); ones on product production and module supply, Eq. (16.11); ones on module assignment to products, Eq. (16.13); ones on product production and module supply, Eq. (16.14); ones on module and product production volume at respective sites, Eq. (16.15); ones on product production volume and product transportation volume, Eq. (16.16); ones on product transportation volume among respective pairs of different sites, Eq. (16.17); ones on product supply and market sales, Eq. (16.19); and ones on product demand and supply on respective site (market), Eq. (16.21) must be considered. Further, Eqs. (16.36) and (16.37) must be considered.

In summary, the simultaneous design problem of module commonalization and supply chain configuration is formulated as a multi-objective, mixed-integer programming problem with 0–1 integer variables and nonnegative real variables.

## 16.4 Optimal Design Techniques for Global Product Family

### 16.4.1 Multi-Objective Optimization

In general, a mixed nonlinear programming problem is combinatorial hard to efficiently find its optimal solution. Further, multi-objective optimization of such a class of problems is more difficult and time consuming to find a set of relevant alternatives.

While the above is general tendency, the optimization problem formulated in the previous section has a specific structure. That is, if the 0–1 integer variables are fixed into either value, the leftover problem becomes a linear programming problem, which can be solved with moderate computation cost, and the objective on profit is only variable over continuous variables. Thus, the original problem is divided into the upper-level subproblem, where the 0–1 integer variables are manipulated under multiple objectives, and the lower-level subproblem, where nonnegative real variables are manipulated under the single objective. A multi-objective genetic algorithm (MOGA) is applied to the former, and a simplex method is used for the latter. This hybridization enables to solve the optimization problem with reasonable computational expense.

Among various MOGA techniques, the neighborhood cultivation GA (NCGA) (Watanabe et al. 2002) is used for the upper-level problem. NCGA is composed of neighborhood crossover that a pair of individuals adjoining in an objective space are selected for crossover, the reference objective is shifted among multiple objectives for avoiding iteration of similar mating, environmental selection is embedded, and so forth.

In the simplex method for the lower-level subproblem, the constraints on real variables under the 0–1 variables are considered by assigning some penalty to the objective Eq. (16.31). If the lower-level problem does not have any feasible solution, its infeasibility is evaluated with stack variables, and the result is fed to the upper-level problem as the penalty. While the genetic algorithm cannot handle constraints usually and the simplex method cannot deal with the quality of infeasible solutions, the above mechanism not only resolves these limitations but also enables efficient optimization computation of the complicated mixed-integer programming problem.

### 16.4.2 *Principal Component Analysis for Clustering Pareto Solutions*

As aforementioned in Sect. 16.2.2, it is necessary that the preferences should be identified from a set of potentially competitive solutions, which are Pareto optimal solutions obtained by the optimization method described in the previous subsection. The design concept exploration techniques are expected to be useful for such a purpose (Obayashi and Sasaki 2003; Fujita et al. 2011). This chapter introduces the principal component analysis (PCA) (Oyama et al. 2009) and its associated clustering algorithm as a data-mining technique for extracting useful information toward such a solution. PCA is a mathematical procedure for converting a set of observations of possibly correlated variables onto a set of values of linearly uncorrelated variables with orthogonal transformation. It is effective for representing complicated observations with less numbers of essential factors. In the data mining with PCA, first the principle components are calculated with a statistical processing technique, and dominant ones are identified by viewing its result. Then, the observations are clustered via their principal scores up to the dominant order of principle components with a clustering algorithm that is named the Ward's method (Ward 1963).

The relationship among the route information and three objective functions is considered to be the essential factor on the characteristics of respective alternatives in the global product family design problem. Thus, when applying the above technique to the problem, 0–1 integer variables representing route information,  $\delta_{j,k}^s$ ,  $\delta_i^{s \rightarrow}$ ,  $\delta_i^{\rightarrow r}$ , are used as the input data to the PCA. 0–1 integer variables are treated as continuous variables under the nature of the PCA mechanism.

## 16.5 Numerical Case Study

### 16.5.1 *Typical Situation of Global Product Family Design*

While the general meaning of global product family is described in Sect. 16.2.1, such a situation can be found in various sectors of manufacturing. Among them, one of automotive industries must be typical (Fujita et al. 2012a). For example, Nissan has introduced the new models of March (called Micra in Europe), a brand name of their world compact cars, in 2010 (Takano 2010). Before this model change, March cars were manufactured in the Japan and England factories, because large markets were around there. However, new models are not produced in Japan, and they are imported from the Thailand factory to Japanese market. The reasons are complicated, but the demands on compact cars are rapidly growing in emerging countries. Nissan decided to manufacture new models at factories in Thailand, India, China, and Mexico. According to an article found in a magazine (Takano 2010), Nissan had revised not only the product architecture and strategies for arranging variants over the world but also the grade of materials such as steel plates for shifting the production bases from developed countries to emerging countries. On the other hand, productions at the Japan factories may concentrate into other brands with different grades. This will be not only because of the distance between factories and major markets but also because such models essentially require high levels of technologies and materials.

### 16.5.2 *Contents of an Example Problem*

With refereeing the situation described in the above subsection, we configure a virtual example problem for demonstrating the effectiveness of optimal design techniques developed in Sect. 16.4 with an image of automotive industries. In the problem, the number of product variants is four. The number of module slots is three. The number of module candidates for individual slots is 2, 2, and 3, respectively. The number of sites for both production and sales is three. Among three sites,  $s^1$  is one of leading industrialized countries such as Japan,  $s^2$  is one of larger developing countries such as China, and  $s^3$  is a model of newly industrialized countries such as Thailand. Their geographical relationship is assumed as well as the real one among those countries. While the model described in Sect. 16.3 includes a lot of parameters to be assigned, their outline can be summarized as shown in Table 16.1. Its some details are assigned as shown in Tables 16.2, 16.3, 16.4, and 16.5.

Besides, the above problem is formulated with 45 0–1 integer variables and 204 real variables. When the optimization computation is performed with 1,000 individuals over 1,000 generations, the optimization computation takes about 30 h on an ordinary personal computer.

**Table 16.1** Outline of the example problem; differences among three sites

Sites	Cost	Quality	Lead time	Market size	Production capacity	Transportation capacity
$s^1$	Higher	Better	Moderate	Larger	Moderate	Larger
$s^2$	Lower	Worse	Longer	Larger	Larger	Larger
$s^3$	Moderate	Moderate	Shorter	Moderate	Smaller	Larger

**Table 16.2** Price of products at respective sites  $p_i^s$  (Yen)

	$s^1$	$s^2$	$s^3$
$P_1$	1,000,000	1,000,000	1,000,000
$P_2$	1,300,000	1,300,000	1,300,000
$P_3$	1,600,000	1,600,000	1,600,000
$P_4$	2,000,000	2,000,000	2,000,000

**Table 16.3** Demand volume of products at respective sites  $D_i^s$  (unit)

	$s^1$	$s^2$	$s^3$
$P_1$	8,000	10,000	5,000
$P_2$	6,000	6,000	4,000
$P_3$	5,000	5,000	3,000
$P_4$	3,000	3,000	2,000

**Table 16.4** Availability of modules for producing products, which means the possibilities of module diversion and commonalization  $\gamma_{j,k \rightarrow i}$

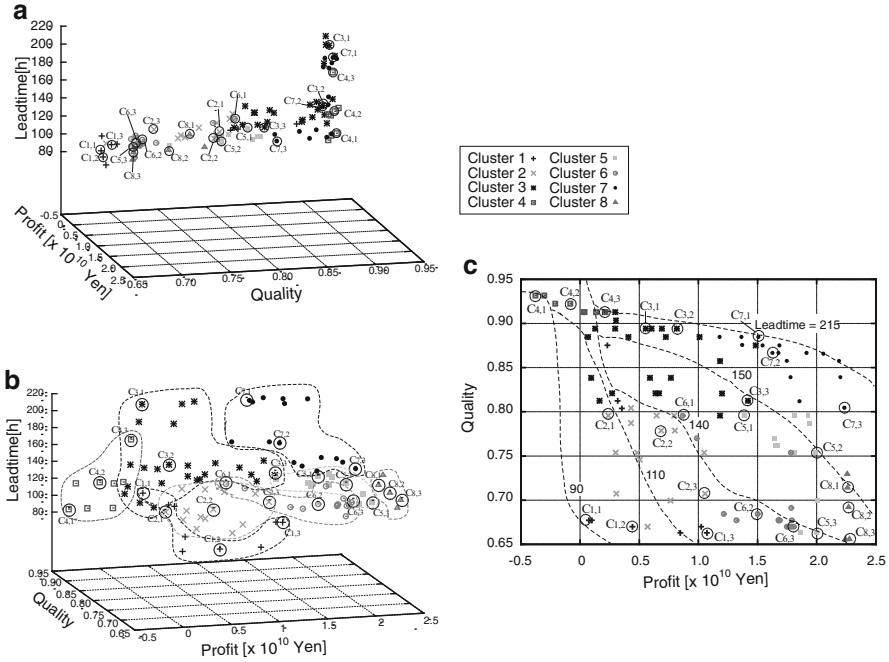
	$m_{1,1}$	$m_{1,2}$	$m_{2,1}$	$m_{2,2}$	$m_{3,1}$	$m_{3,2}$	$m_{3,3}$
$P_1$	1	0	1	0	1	1	1
$P_2$	0	1	1	0	1	1	1
$P_3$	1	1	0	1	0	1	0
$P_4$	0	1	0	1	0	0	1

**Table 16.5** Production and sales capability of respective sites

		$s^1$	$s^2$	$s^3$
$W^s$	(unit)	100,000	150,000	80,000
$C_S^s$	( $\times 10^6$ Yen)	3,600	2,400	3,000
$C_{PS}^s$	( $\times 10^6$ Yen)	1,000	1,000	1,000

### 16.5.3 Multi-Objective Optimization

The optimization method described in Sect. 16.4.1 is applied to the example problem. Regarding the number of tentative Pareto optimal solutions, the number is gradually increased from about 70 individuals at the 100th generation to about



**Fig. 16.3** Pareto optimal solutions and their clustering. (a) Cross-sectional perspective view. (b) Projected perspective view. (c) Plot of solutions to quality-profit section

90 individuals at the 400th to 500th generations, and finally 105 Pareto optimal solutions are obtained at the 1,000th generation.

### 16.5.4 Clustering of Pareto Solutions

Figure 16.3 shows the distribution of such Pareto solutions and their clustering with the PCA method. The solutions are categorized into eight clusters; respective plots are marked with different symbols by categorized clusters. Part (a) shows a cross-sectional perspective view of Pareto frontier surface, Part (b) shows their projected perspective view in the orthogonal angle of Part (a), Part (c) shows their projection to quality-profit section. The mathematical symbol  $C_{i,j}$  indicates that it is the  $j$ th Pareto solution of the  $i$ th cluster. In the figure, it is confirmed that quality, profit, and lead time are traded off indeed as common sense indicates. For example, the solutions in Cluster 1 are superior in lead time but inferior in quality and profit. Ones in Cluster 4 are superior in quality but inferior in profit. Ones in Cluster 8 are superior in profit but inferior in quality.

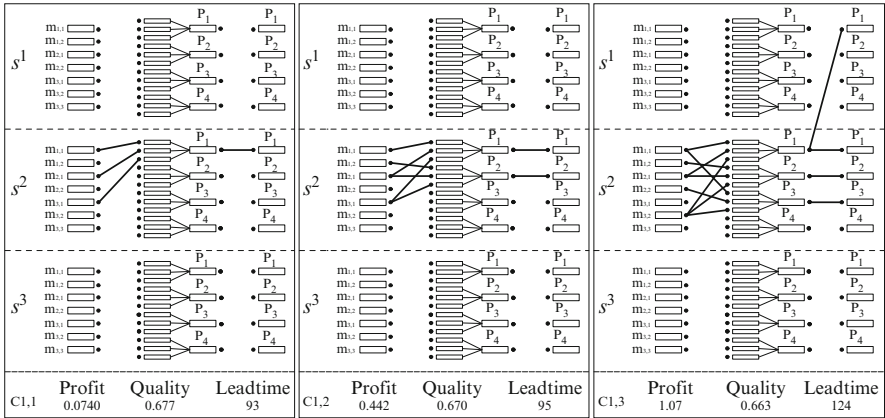


Fig. 16.4 Representative solutions in Cluster 1

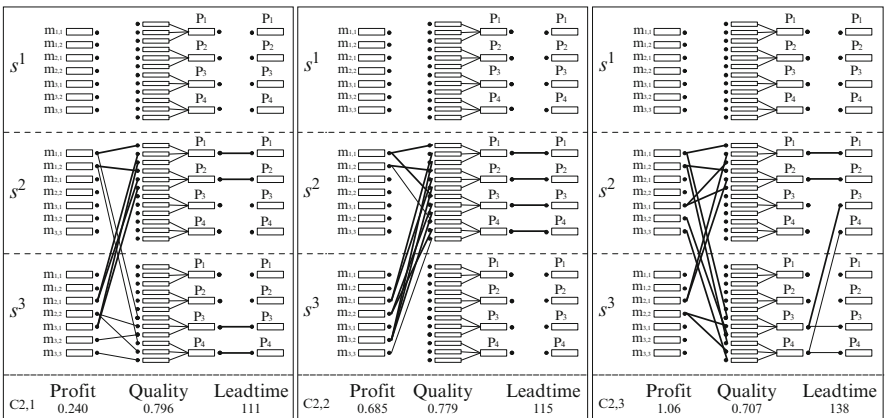


Fig. 16.5 Representative solutions in Cluster 2

### 16.5.5 Design Concept Exploration

Figures 16.4, 16.5, 16.6, 16.7, 16.8, 16.9, 16.10, and 16.11 show the contents of the representative solution of respective clusters. In each rectangle of each figure, icons on module production are arranged in the left column, ones on product assembly are arranged in the center column, ones on product sales are arranged in the right column, and horizontal broken lines distinct them into ones at different sites. An empty icon means that the corresponding module or product is not produced or sold there. The lines between module production and product assembly and between product assembly and product sales represent the transportation volume at the respective routes with three levels of their thickness. Under these icons and lines, the solution index and the values of three objectives are shown.



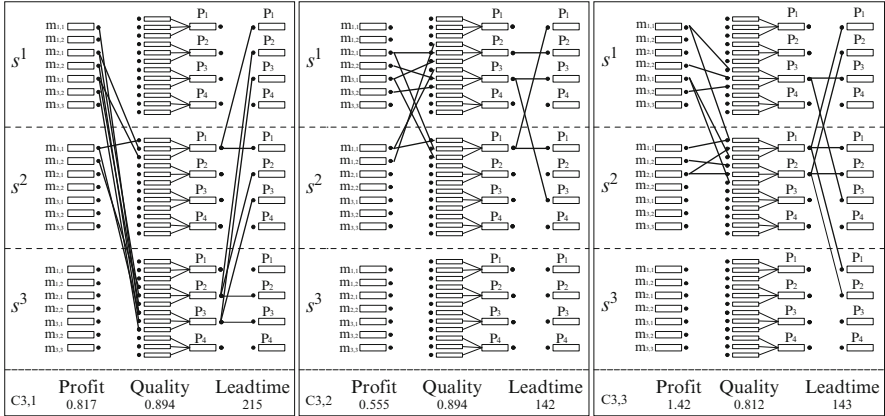


Fig. 16.6 Representative solutions in Cluster 3

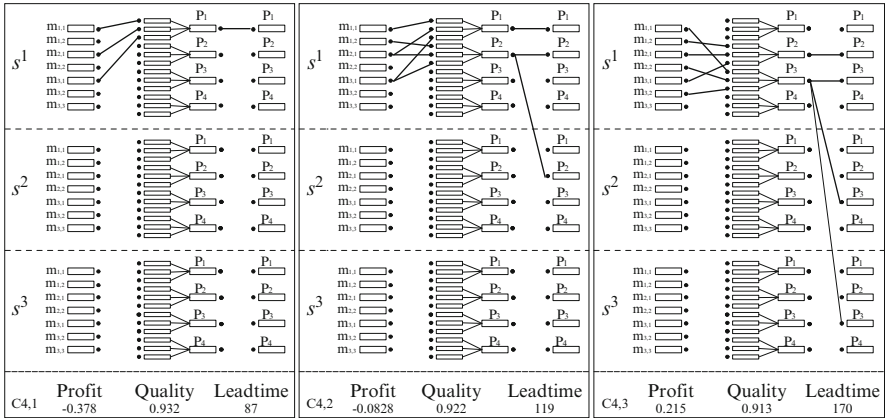


Fig. 16.7 Representative solutions in Cluster 4

The following explains the characteristics of respective clusters and associated interpretations. In their heading, the figure number indicates the figure where the contents of representative Pareto solutions within each cluster are illustrated. The numbers in parentheses are the ranking of three objective functions among eight clusters.

Cluster 1 (Fig. 16.4)—Production and assembly are performed at only Site  $s^2$  or at Sites  $s^1$  and  $s^2$ . Therefore, the scale of product deployment is quite small. While lead time is shorter, profit and quality are inferior ( $Profit : 7, Q_{min} : 6, L_{max} : 1$ ).

Cluster 2 (Fig. 16.5)—Production and assembly are performed at Sites  $s^2$  and  $s^3$ . Therefore, the scale of product deployment is rather small. Since modules and products are transported between Sites  $s^2$  and  $s^3$ , lead time is better, where the

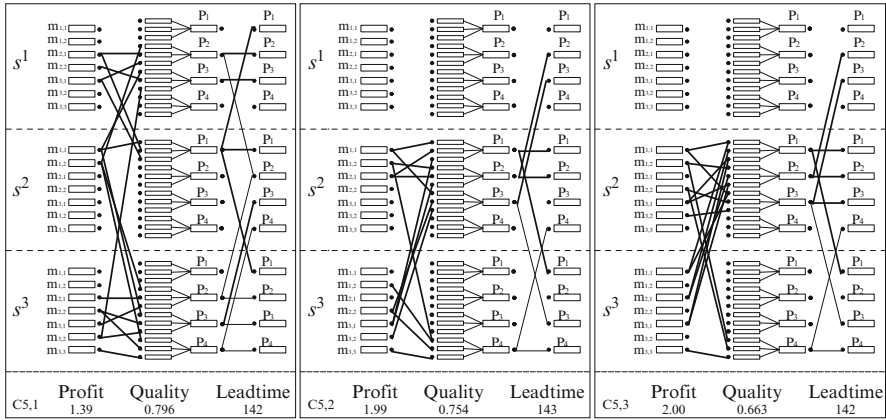


Fig. 16.8 Representative solutions in Cluster 5

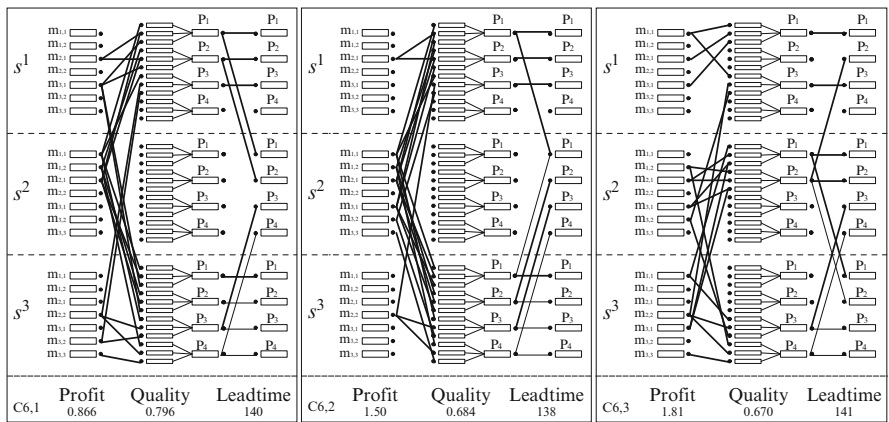


Fig. 16.9 Representative solutions in Cluster 6

distance between Sites  $s^1$  and  $s^2$  and one between Sites  $s^2$  and  $s^3$  are about half of one of Sites  $s^1$  and  $s^3$ . Quality is moderate and profit is not so good ( $Profit : 6, Q_{min} : 4, L_{max} : 3$ ).

Cluster 3 (Fig. 16.6)—Few production are at Site  $s^3$  and the scale of product deployment is small. The module production is arranged with a high regard for quality, as Modules  $m_2$  and  $m_3$  that are difficult to produce are produced at Site  $s^1$ , and Module  $m_1$  that are easy to produce are produced at Site  $s^2$ . Thus, while quality is better, profit is not so good. Further, when assembly at Site  $s^3$  is performed supplementally, modules are transported from Site  $s^1$  to Site  $s^3$ , and lead time becomes longer ( $Profit : 5, Q_{min} : 2, L_{max} : 7$ ).

Cluster 4 (Fig. 16.7)—Production and assembly are performed only at Site  $s^1$ . Quality is the best, and lead time is very short. On the other hand, the scale of product deployment is quite small, and profit is worst ( $Profit : 8, Q_{min} : 1, L_{max} : 2$ ).

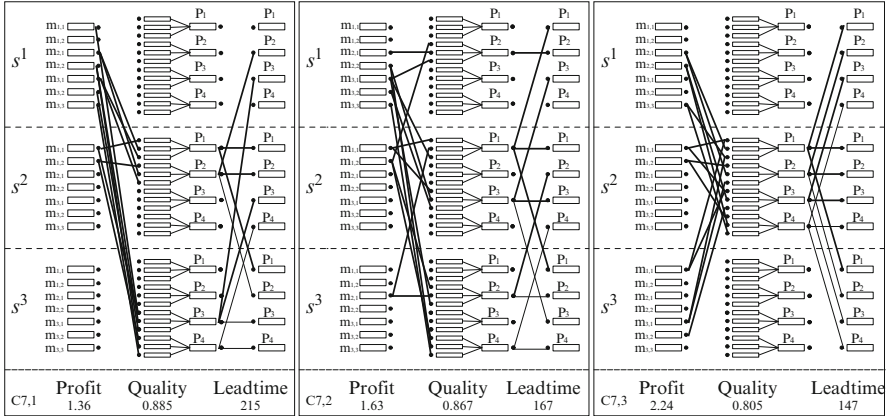


Fig. 16.10 Representative solutions in Cluster 7

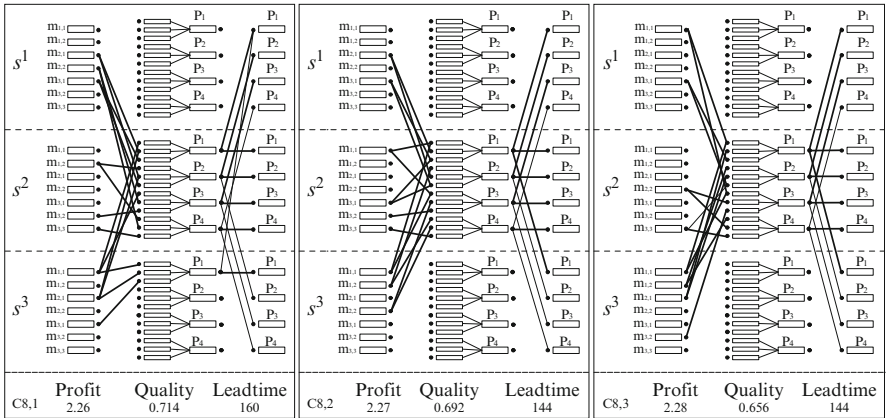


Fig. 16.11 Representative solutions in Cluster 8

That is, it is necessary to enlarge the scale of product deployment for increasing profit. However, under the constraints on production volumes, it becomes indispensable to utilize all sites for module production and product assembly. Since the product deployment depends on export and import very much, lead time becomes deteriorated in general. As production and assembly are performed at all sites, the products that are composed of modules produced at Site  $s^2$  are increased anyhow. Consequently, quality has become lower, as the objective is the worst case of quality among all products. The following clusters can be categorized into ones with such a viewpoint.

Cluster 5 (Fig. 16.8)—The scale of product deployment is large. Modules are produced with a high regard for quality fundamentally. Quality is not so good, because some are produced at Site  $s^2$ . However, quality is better among the last four

clusters that have the large scale of product deployment. Lead time is better, as export and import between Sites  $s^1$  and  $s^3$  (*Profit* : 2,  $Q_{min}$  : 5,  $L_{max}$  : 5).

Cluster 6 (Fig. 16.9)—While the scale of product deployment is large, assembly at Site  $s^2$  is not so large, but many modules are produced at Site  $s^2$ . Thus, quality is not so good, while profit and lead time are moderate. Among the later four clusters, profit is the worst due to smaller volume of assembly at Site  $s^2$  (*Profit* : 4,  $Q_{min}$  : 7,  $L_{max}$  : 4).

Cluster 7 (Fig. 16.10)—Under the large-scale product deployment, many modules are produced at Site  $s^1$ , and many products are assembled at Site  $s^2$ . Regarding the activities at Site  $s^3$ , there are two cases that module production is major and that product assembly is major. In both cases, quality is better, because modules produced at Site  $s^1$  are dominant. Profit is better, since module production that can be performed with relatively low cost is executed at Site  $s^1$ , and since product assembly that must be performed with high cost is executed at Site  $s^2$ . Lead time is the worst, since modules are transported from Site  $s^1$  to Site  $s^3$  and products are transported from Site  $s^3$  to Site  $s^1$ . This tendency does not depend on the scale of product deployment within these clusters (*Profit* : 3,  $Q_{min}$  : 3,  $L_{max}$  : 8).

Cluster 8 (Fig. 16.11)—The scale of product deployment is large, products are assembled at Site  $s^2$ , and then exported to all sites. Since module production is arranged with a high regard for cost, profit is the best, but quality is the worst, while lead time is worse. However, as the scale of product deployment becomes larger within this cluster, lead time becomes moderate, since there is no export or import between Sites  $s^1$  and  $s^3$  (*Profit* : 1,  $Q_{min}$  : 8,  $L_{max}$  : 6).

The above interpretations of representative competitive solutions are enabled through lump generation of Pareto solutions by multi-objective genetic algorithm and their clustering by a PCA-based data-mining technique. Since those interpretations correspond to conceptual meaning of concrete solutions and major modes of concrete trade-offs among quality, cost, delivery, etc., they are effective for concept-level decision making of simultaneous design of module commonalization and supply chain configuration. That is, it can be recognized that the scenario mentioned in Sect. 16.2.2 is realized through the proposed optimal design techniques in designing a global product family.

## 16.6 Concluding Remarks

This chapter described a multi-objective formulation of a class of global product family design problem, in which module commonalization and supply chain configuration are simultaneously determined and its optimization algorithm by combining the neighborhood cultivation genetic algorithm and the simplex method. Further, this chapter explores design concepts underlying a set of Pareto optimal

solutions obtained by using the principal component analysis-based clustering technique. The demonstrated results indicate the possibility and promise on the application of multi-objective optimization and design concept exploration to the global product family design. While outcomes are restricted by various conditions, the mathematical model of this chapter is expected to be foundations for rationally exploring the excellence of global product family design.

**Acknowledgement** The author acknowledges that computer programming and computation of optimization examples were done by Ken Nasu, who was formerly a graduate student of Osaka University, and Yuma Ito, who is currently a graduate student of Osaka University.

## References

- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
- Fujita K, Sakaguchi H, Akagi S (1999) Product variety deployment and its optimization under modular architecture and module commonalization. In: Proceedings of the 1999 ASME design engineering technical conferences, Paper No. DETC99/DFM-8923
- Fujita K, Muraoka M, Mistunaka A, Nomaguchi Y (2011) Preliminary study on design concept exploration of truss structures by multi-objective optimization and self-organizing map. In: Proceedings of 9th world congress on structural and multidisciplinary optimization (WCSMO-9), Paper Code 361\_2
- Fujita K, Amaya H, Akai R (2012a) Mathematical model for simultaneous design of module commonalization and supply chain configuration toward global product family. *J Intell Manuf*. doi:[10.1007/s10845-012-0641-x](https://doi.org/10.1007/s10845-012-0641-x)
- Fujita K, Nasu K, Ito Y, Nomaguchi Y (2012b) Global product family design: multi-objective optimization and design concept exploration. In: Proceedings of the 2012 ASME design engineering technical conferences and computers and information in engineering conference, Paper No. DETC2012-70858
- Jiao J, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18(1):5–29
- Obayashi S, Sasaki D (2003) Visualization and data mining of Pareto solutions using self-organizing map. In: Proceedings of 2nd international conference on evolutionary multi-criterion optimization, pp 796–809
- Oyama A, Verburg P, Nonomura T, Fujii K (2009) Flow data mining of Pareto-optimal airfoils using proper orthogonal decomposition. In: Proceedings of annual conference of the Japan Society for computational engineering and science (JSCES), vol 14, pp 123–126 (in Japanese)
- Pine BJ (1993) Mass customization: the New Frontier in business competition. Harvard Business School Press, Boston, MA
- Sato K, Wakao S (2010) Data mining method for battery operation optimization in photovoltaics. *IEEE Trans Power Energy* 130(3):313–319
- Shimizu E, Isogai K, Obayashi S (2008) Multiobjective design study of a flapping wing power generator. *ASME J Fluids Eng* 130(2):0211041–0211048
- Simchi-Levi D, Kaminsky P, Simchi-Levi E (1999) Designing and managing the supply chain: concepts, strategies, and cases. McGraw-Hill/Irwin, New York, NY
- Simpson TW, Siddique Z, Jiao J (2005) Product platform and product family design: method and applications. Springer, New York, NY
- Takano N (2010) New MARCH that will be mass-produced at an emerging country, Nikkei Monozukuri, Oct 2010 Issue, Nikkei Business Publications (in Japanese)

- Ulrich E (1995) The role of product architecture in the manufacturing firm. *Res Policy* 24(3):419–440
- Ward JH (1963) Hierarchical grouping to optimize an objective function. *J Am Statist Assoc* 58(301):236–244
- Watanabe S, Hiroyasu T, Miki M (2002) Neighborhood cultivation genetic algorithm for multi-objective optimization problems. In: *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL-2002)*, pp 198–202

# Chapter 17

## Architecture-Centric Design Approach for Multidisciplinary Product Development

A.A. Alvarez Cabrera, H. Komoto, T.J. van Beek, and T. Tomiyama

**Abstract** Managing complexity is a crucial task during the development process of multidisciplinary complex products. To achieve an efficient and effective development process of such a product, all the stakeholders must maintain a common understanding of the system and mutually linked detailed information of the product. This chapter proposes system architecture as a concept wider than product architecture, which provides such an overview as well as information that links various detailed information about the product. System architecture includes not only structural elements and relations among them but also functions, behaviors represented by working principles, and a variety of requirements. The working principles are modeled with physical phenomena and the involved parameters and relations among those parameters (e.g., equations). This chapter presents three prototype tools for system architecting illustrated with examples.

### 17.1 Introduction

In a previous edition of this book (Simpson et al. 2006), a platform is defined as the collection of common elements (not necessarily product parts) which allow efficiently developing and launching a series of products. The authors would like to highlight two key concepts in such a definition:

---

A.A. Alvarez Cabrera • T.J. van Beek  
Delft University of Technology, Delft, The Netherlands

H. Komoto  
National Institute of Advanced Industrial Science and Technology, Tsukuba, Ibaraki, Japan

T. Tomiyama (✉)  
Cranfield University, Cranfield, UK  
e-mail: [t.tomiyama@cranfield.ac.uk](mailto:t.tomiyama@cranfield.ac.uk)

- Deciding on commonality between products is at the core of platforming. Platforming entails providing a generic umbrella to capture and utilize commonality in products including anchoring to a common process structure (Martinez et al. 2000).
- Evaluating the efficiency of a platform is necessary to make decisions about which elements should be made common through the platform. Growing stakeholder awareness about several aspects of development (e.g., costs) increases the ability to make well-informed decisions [the specific case of designers and cost is presented in Simpson et al. (2006)].

On the one hand, answering the question of what should be common in a platform is already a difficult matter (the purpose of this book). On the other hand, even outside the scope of platform development, modern products are increasingly becoming complex for a variety of reasons (Avigad et al. 2003; Craig et al. 1999; Van Amerongen 2003; Xu and Zou 2007) including:

- The customer requirements are becoming complex reflecting severe competition among manufacturers. These requirements may involve technically new functions such as adaptability.
- The sheer “size” involved in modern product design is enormous. Not only the size but also the number of stakeholders also increased.
- Increasingly more disciplines are involved, which requires careful consideration of interactions coming from integrating subsystems.

Consequently, even products that look simple when considering the number of parts may involve much complexity because of the scale and number of the involved phenomena. Further complexity adds due to the production context, e.g., mass customization, distributed development teams, and development of product platforms. Dealing with such complexity, and in particular with that one originated from multidisciplinary, has been the main driver of the proposals presented through this chapter. These points are rarely addressed by other authors in an applied level.

Because of the complexity-related issues described above, platform development (or platforming) must be properly guided by a method and supported by tools. The work presented in this chapter is an attempt to establish them. Among others, however, the chapter focuses on “architecture” and “architecting” that are tightly related to platform development.

“Product architecture” is traditionally understood as the mapping of function and structure or as the fundamental structure of the product (Simpson et al. 2006, Chap. 13, p. 308). The previous edition of this book also shows the direct influence of the product architecture in activities including costing, measuring commonality, platform planning or extent evaluation, product family planning, as well as an integral part in supporting tools for platform development. Therefore, product architecture and product platforms are inseparably and mutually influencing each other.



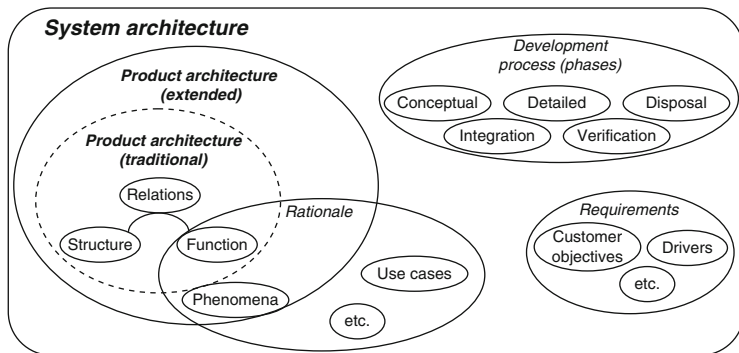
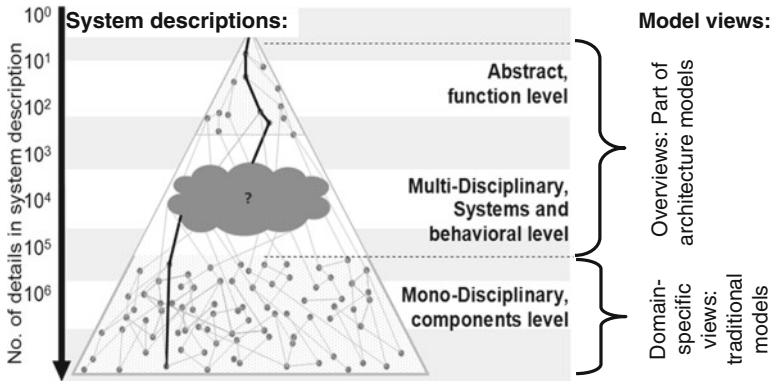


Fig. 17.1 Elements in product architecture and system architecture

The proposals in this chapter consider using models of the “system architecture<sup>1</sup>” as the backbone for tooling which can effectively support platforming by addressing the key issues of determining commonality and evaluating efficiency (highlighted at the beginning of this chapter). With that in mind, it is necessary to provide a more precise definition of what is meant here with terms such as product and system architecture. As shown in Fig. 17.1, product architecture is defined here as an extended concept from the traditional definition mentioned before (consisting largely of connectivity information). In this chapter, product architecture not only considers the structural elements, their relations, and functionality but also takes into account part of the rationale behind the product working principles which explain its behavior. This is considered by modeling the phenomena through the involved parameters and relations among those parameters (e.g., equations). Beyond the scope of product architecture, the system architecture also considers elements around the product and its development, such as requirements and the different development phases, which provide a context responsible for driving and steering the development process.

Having defined the basic concepts, it is necessary to stress that architecture is only a conceptual construct, and, as it is the case for most other concepts, it can only be used effectively when it is modeled. Modeling allows working with the information by enabling to manipulate and to share it, modeling allows building tools. Examples of models can be found within most domains such as free-body diagrams and equations for mechanics, circuit diagrams for electric and electronics, or value models for economics. Such models are referred through this chapter as domain specific. Similarly, two basic models of the architecture are used through this chapter. The first one is the Function-Behavior-State (FBS) model (Umeda et al. 1996), which focuses on representing the product architecture (i.e., the extended

<sup>1</sup>In this chapter, the term of “system architecture” is used as a wider concept than just “product architecture” to signify that the target is complex multidisciplinary systems.



**Fig. 17.2** Model views in product (platform) development, adapted from Muller (2007) and Tomiyama et al. (2007)

definition provided here). The second one has been coined as the Architecture Model (AM) (Alvarez Cabrera 2011), which has been conceived as an extension of the FBS model aiming to represent the system architecture.

Another important point to define is that, due to the complexity of the models discussed above, special attention has to be paid to the task of managing the information at any given point in time. The experience of the authors has shown that a big part of this task can be handled by using appropriate views which balance between providing an overview of the system (to maintain context) while making available the relevant information for a specific discussion (a particular analysis or evaluation). As depicted in Fig. 17.2, views covering both levels are necessary to address the descriptions made at different levels of detail. On the one hand, domain-specific views are tightly related to the model they relate to, and thus become defined (as the relevant the types of information and their relations) mostly by such models. On the other hand, such a straightforward definition is not yet found for overviews (this is partially addressed here).

The concept of views in models of the system architecture (called also system architecture descriptions) is presented also in the ISO/IEC 42010 standard (IEEE 2000). The standard considers the architecture model as the aggregation of multiple-shared views (see Fig. 17.3) addressing the concerns of relevant stakeholders. The term “stakeholders” envelops the group of actors involved in a development process, including customers, designers, engineers, managers, and even their working tools. As a result, and confirming the reasoning made earlier, these views must refer to the objects in the domain-specific models that the stakeholders use for development and design.

This chapter tries to provide tools to integrate domain-specific methods by integrating the execution, results, and views they can provide. The architectural model has as goals to provide an overview of the system by facilitating common understanding and to work as a formal basis to support development activities

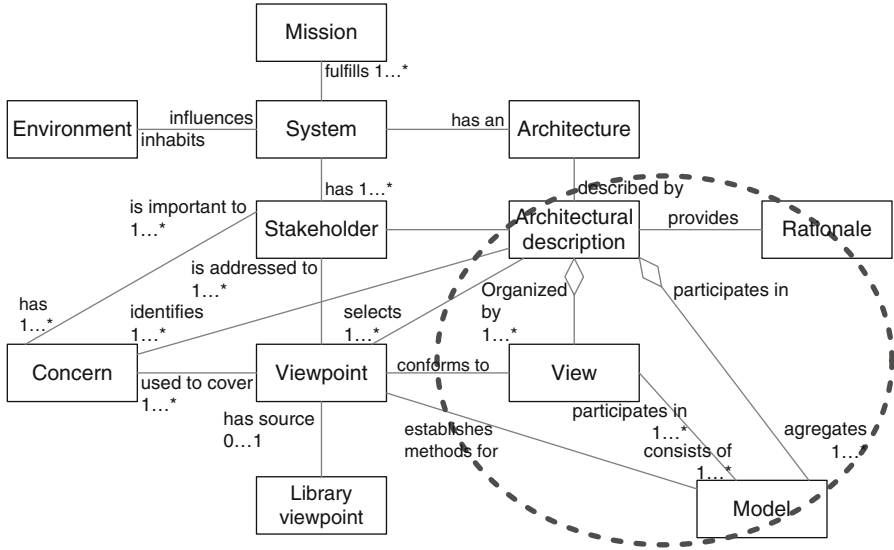


Fig. 17.3 Conceptual model of an architectural description, adapted from IEEE (2000)

including the architecting activity itself (i.e., defining a system architecture). In this context, the impact of system architecture models on product development becomes deeper as it directly reflects on the capability of efficiently evaluating design alternatives by providing the necessary overview and pointing out to the specific models that hold domain-specific information.

Designing platforms may take place simultaneously with the definition of the system architecture, implying that not much quantitative and precise information may be available at such a design stage. With that in mind, performing analyses to compare platform designs with less detailed (but formal) architecture-level information becomes a necessity and will be demonstrated with an example later. It is worth noting that this chapter does not deal with how to identify or create any given type of architecture (e.g., modular or integral) and that the presented techniques and models help to consider and evaluate different aspects related to platform-based development and architecting.

The rest of this chapter is structured in three sections dedicated to show examples following the common line of how to support development activities by using models of the architecture. However, the examples do not share case studies as they are a compilation of past work of the authors. The first section shows examples of our efforts to support the architecting activity itself (i.e., architecting) before explicitly considering other design activities. The second section presents specific techniques to aid crucial tasks for architecting: definition of overviews and interfaces for modularization. Those two sections make direct use of models of the product architecture (i.e., the extended definition provided here), manipulating

them as part of the development process. The third section is focused on explaining how system architectures can be formally captured to accomplish the goals of providing overview and supporting analysis and evaluation of alternatives described in multiple domain-specific models while following a truly model-based paradigm.

There is much related literature discussing topics such as design support, architecting, modularization, and design automation. However, for the sake of being practical and direct, this chapter does not aim to provide a review of such literature. References to related work are brought when touched directly and can be found in more detail in the cited original publications from which the three main examples are taken. The reader can also find useful review material in Erden et al. (2008).

## **17.2 Architecting Activity in Product Development and Its Relevance to Product Platform Development**

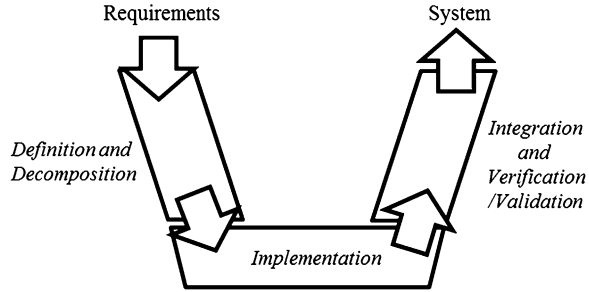
This section briefly explains the architecting activity using a development framework based on a model of the product architecture. Particularly, this section discusses the development of a product platform as a part of the architecting activity. Furthermore, computational supports used in the architecting activity are briefly explained.

### ***17.2.1 Architecting Activity in Product Development***

Figure 17.4 shows a product development framework from the perspective of systems engineering known as the V-model (Forsberg and Mooz 1991). The V-model consists of three phases, the system decomposition phase, the implementation phase, and the system integration and verification/validation phase. The first phase concerns decomposition of design problems. In the middle phase, corresponding solutions to the decomposed design problems are found. The last phase concerns synthesis of design solutions. The architecting activity discussed in the section is performed at the first phase and consists of three major tasks (Komoto and Tomiyama 2011):

- Identification of requirements to be translated to system-level specifications (functions) of products.
- Hierarchical decomposition of the system-level specifications to specifications of subsystems and eventually to components.
- Definition of behaviors and structure of these subsystems and their interfaces to meet the corresponding specifications.

**Fig. 17.4** Product development based on the V-model (Forsberg and Mooz 1991) highlighting the phase addressed in this section



After performing these tasks, components and subsystems are designed in detail (the implementation phase) followed by validation, verification, and integration of designed components and subsystems at the latter phase of product development.

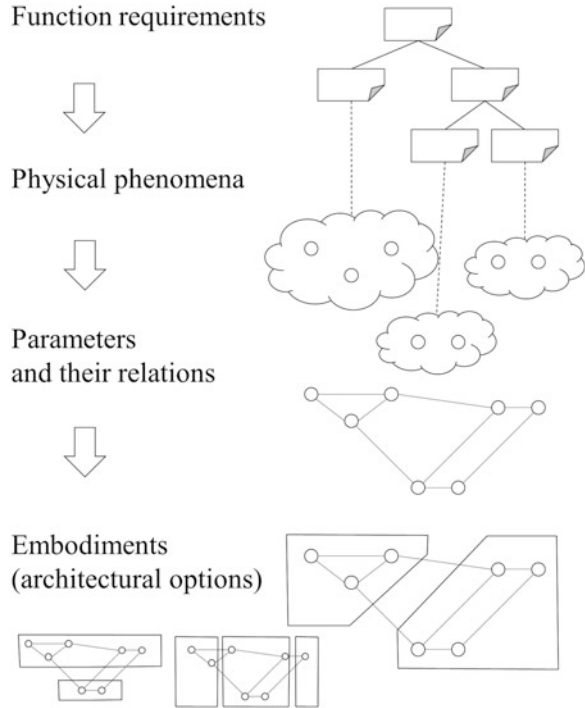
Engineers in charge of these tasks are often called system architects. The system architect performs these tasks in cooperation with domain experts (e.g., mechanical, control, and software engineers). The architect should understand the properties of such embodiments based on similar subsystems developed in past designs and standard components in handbooks from specific aspects. Using their design experience, the architect can envision the behavior of embodiments in a given environment. He/she collects and abstracts the knowledge of domain experts about embodiments and defines relations (interfaces) among them so they satisfy the overall requirements as a whole.

The system architect encounters the following challenges in the architecting activity:

- In the process of hierarchical decomposition of the system-level specifications into lower-level specifications, the system architect should maintain consistency among the descriptions with different terms used in the different levels of hierarchy and domains.
- The system architect may not be able to find embodiments satisfying specific requirements from existing subsystems. In this case, he/she has to design new embodiment satisfying such requirements with the help of domain experts.
- Even though he/she can identify embodiments satisfying all requirements, there are a number of possible integrations of embodiments in terms of interfaces among them. Therefore, systematic evaluation of possible integrations is necessary before an integrated system is actually built.

The first and third problems have been studied in the field of function modeling (e.g., Umeda et al. 1996) and modularization (e.g., Erixon et al. 1996), respectively. For instance, the study of function modeling suggested the use of abstract (symbolic) descriptions to represent a design concept across engineering domains in function development. Clustering methods have been applied to evaluation and optimization of the modular architecture of products. However, few studies have tackled the second problem. Next, the second problem in architecting activity is further analyzed while considering the development of a product platform.

**Fig. 17.5** Architecting activity in product development



### ***17.2.2 Product Platform Development in Architecting Activity***

As introduced in the previous subsection (further detailed in Fig. 17.5), the system architect first defines the function requirements of products in a family. Some function requirements are assigned as common in family members, while others are uniquely given to one product type. Based on the commonality and uniqueness of function requirements defined across product types, the system architect defines the modular architecture of products. These modules form the basis of the product platform. As explained above, he/she may not be able to find appropriate embodiments to satisfy function requirements. Missing embodiments can be either common parts or parts dedicated for a specific product. In both cases, it is inevitable to design missing embodiments during the product development.

The previous work of the authors has developed a method to perform architecting activity without requiring the primary definition of embodiments or building blocks (Komoto and Tomiyama 2011). Building blocks are, e.g., machine elements, established components, and mechanisms in mechanical design. In control design, fundamental building blocks are represented as block diagrams including sensors, actuators, and controllers. In software design, they can be subroutines and functions. In case of the development of a product platform, building blocks serve as common modules or dedicated structural elements.

Figure 17.5 shows the architecting activity in product development, which includes a procedure necessary when existing embodiments do not satisfy all requirements. The system architect identifies physical phenomena realizing a desired system behavior (that satisfies a function requirement) together with parameters associated with the phenomena (e.g., parameters in an equation characterizing a physical phenomenon). These parameters constitute a network that can be divided into clusters. While a cluster indicates an embodiment of a component or a subsystem, a set of clusters as a whole defines a module. The system architect compares architectural options regarding the organization of clusters. Multiple architectural options are derived from the network of parameters obtained from a given set of physical phenomena. Each physical phenomenon is defined with a subnet of entities, their parameters, and relations among the parameters representing the physical phenomenon (e.g., the governing equation derived from the physical phenomena). The definition is stored in the knowledge base of computer tools used by the system architect (such as SA-CAD explained later). An architectural option is defined as a direct sum decomposition of the parameter set. By exhaustively searching possible direct sum decompositions, all possible architectural options can be computed (Komoto and Tomiyama 2011). The algorithm is explained at the end of this subsection.

The architecting activity in Fig. 17.5 becomes more complex in case of the development of a product platform (as opposed to a single product). For instance, some subsystems and components can be treated as common parts in the product platform. Such common parts should be determined at the architecting activity of products, which include function requirements realized by the common parts in parallel.

### 17.2.3 Computational Support

The authors have developed a prototype Systems Architecting CAD (SA-CAD) tool supporting the architecting activity (Komoto and Tomiyama 2010, 2011). The types of support offered by the system are:

- Hierarchical decomposition of function requirements and definition of structural and behavioral descriptions based on the FBS modeling (Umeda et al. 1996).
- Generation of architectural options at any level of function hierarchy (i.e., SA-CAD is used to define the top-level decomposition of the system to satisfy the major function requirements as well as the decomposition of a subsystem to satisfy the local function requirements of the subsystem).
- Visualization of a product model using diverse (functional, behavioral, geometric, and parameter level) aspect modelers.
- Consistency management of design information defined with the above aspect modelers (Komoto and Tomiyama 2010).

SA-CAD is equipped with four aspect modelers, the FBS modeler, the physical process modeler, the geometric modeler, and the parameter network modeler (Fig. 17.6). Using the FBS modeler, a product is modeled in terms of abstract



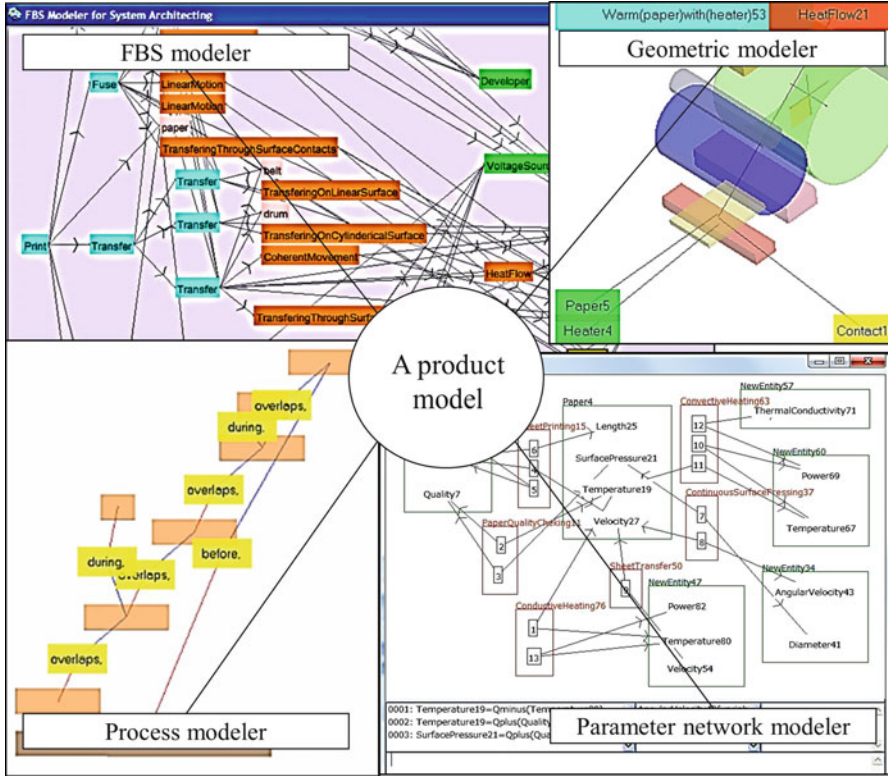


Fig. 17.6 SA-CAD to support architecting activity

concepts such as functions, physical phenomena, relations, and entities (i.e., the elements of product architecture). The physical process modeler defines temporal relations among physical phenomena (adding more detail to phenomena descriptions). The geometric modeler handles geometric relations among entities (simplifying the definition of the abundant geometric relations). The parameter network modeler is used to define parameters of entities as well as relations among the parameters based on the definition of physical phenomena (facilitating phenomena definition and visualization). In architecting activity, the parameter network modeler plays a major role, while the other aspect modelers are used to help the system architect understand the model of a product from functional, behavioral, and structural aspects.

Figure 17.7 depicts an architecting case of an air-conditioning system with screenshots of the parameter network modeler. The detailed description of the case is beyond the scope of this section. Figure 17.7a shows four blocks with a hierarchical structure. These blocks indicate the air-conditioning system (*airConditioningSystem*) and other entities (*world*, *room*, and *human*). Relations between the parameters of the air-conditioning system and those of the other entities should be



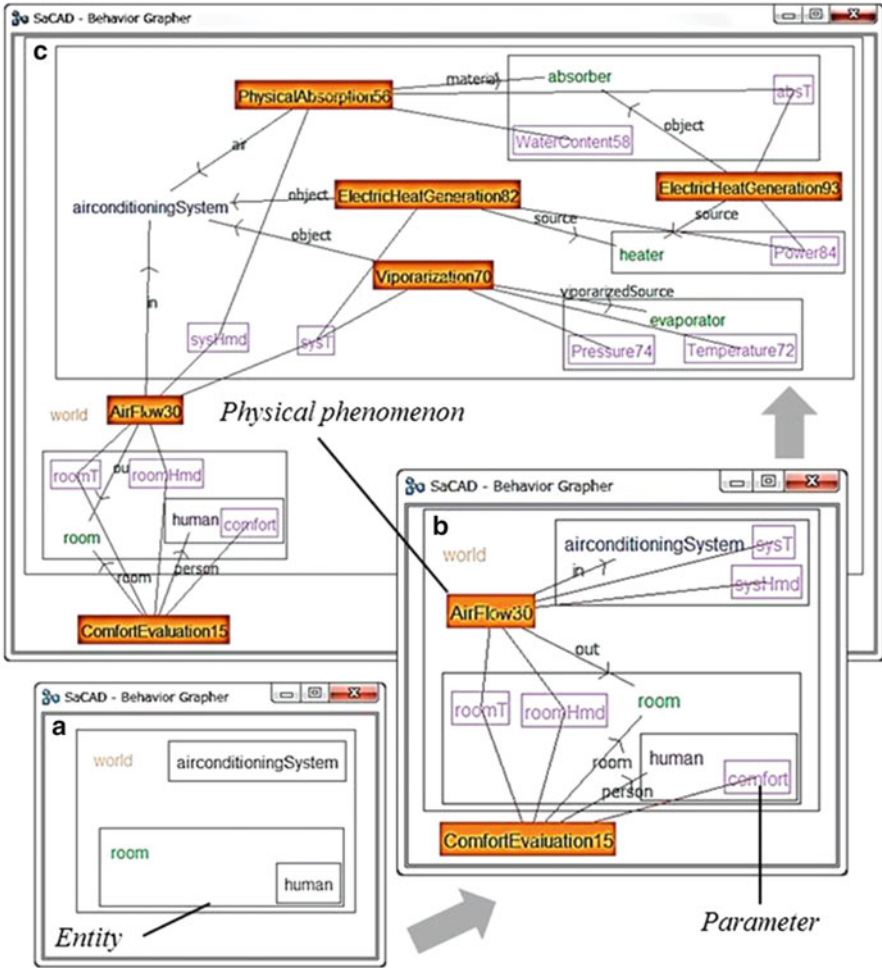


Fig. 17.7 Development of product architecture with SA-CAD

specified so that the air-conditioning system can adapt itself to the variations in the parameter values (i.e., state) of the other entities. At the beginning of the architecting case, the air-conditioning system does not possess subsystems and their parameters.

Figure 17.7b shows a part of the architecting activity in which some parameters of the air-conditioning system were defined based on the initial requirement of the system. The initial functional requirement of the system was “to increase the comfort of *human*,” in which the comfort was treated as a parameter of *human*. The initial requirement is evaluated according to the temperature and humidity of *room* (*roomT* and *roomHmd*).

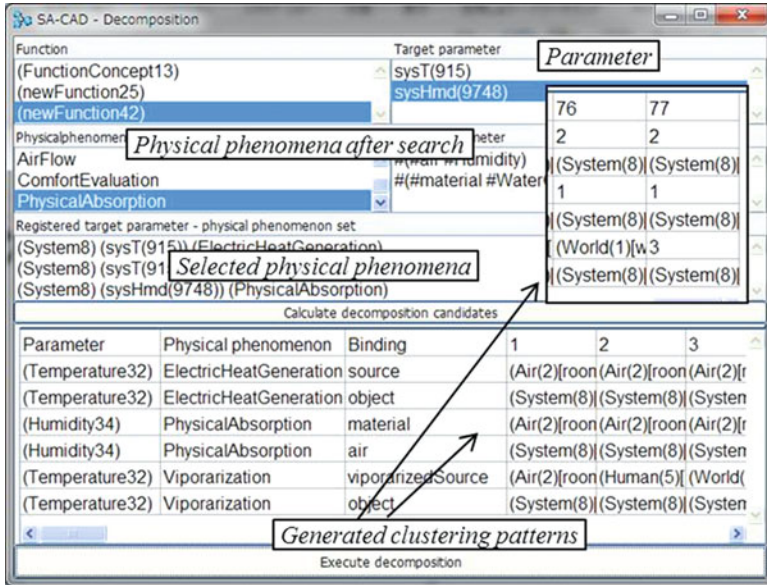


Fig. 17.8 Physical phenomena search and clustering pattern generation on SA-CAD

The decomposition was defined by analyzing how *human* perceived the comfort under the given surroundings (i.e., *room* in *world*). SA-CAD supported the architect by searching and displaying physical phenomena that can influence these parameters. As shown in Fig. 17.7b, *Airflow* was the selected physical phenomenon to influence the parameters. When the physical phenomenon is instantiated, parameter relations defined in the selected phenomenon are also instantiated. In the case, the temperature and humidity of *airConditioningSystem*(*sysT* and *sysHmd*) were instantiated.

Figure 17.7c shows subsystems of the air-conditioning system and their parameters defined with the following procedure supported by SA-CAD. First, SA-CAD searches physical phenomena influencing *roomT* and *roomHmd*, which have been identified in advance. Among several candidate physical phenomena (e.g., *airflow*), the architect chose *Vipolarization* and *ElectricHeatGeneration* to influence the value of *sysT* and *PhysicalAbsorption* to influence the value of *sysHmd*.

SA-CAD generates architecting options by clustering the parameters defined in the selected physical phenomena. Figure 17.8 depicts a screenshot of the interface of SA-CAD in which the architect has selected a set of physical phenomena and a clustering pattern among possible clustering patterns (i.e., physical decompositions). In the case, the architect selected a clustering pattern in which three independent subsystems (*absorber*, *heater*, and *evaporator*) are introduced so that they individually cause the physical phenomena. The clustering pattern is one of 77 patterns generated by SA-CAD. In Fig. 17.8, the selected clustering pattern 77 includes indices 1, 2, and 3, which represent three independent subsystems instantiated at this stage.

Furthermore, the architect considered another *ElectricHeatGeneration* so that the temperature of *absorber* (*absT*), which influences the value of *sysHmd*, can be adjustable with *heater*. SA-CAD helps the architect define the additional roles of existing subsystems with the introduction of new physical phenomena. The support also uses the procedure to search physical phenomena and generate clustering patterns described in the previous paragraph.

## 17.3 Modularization and Definition of Interfaces

Modularization is the core of many supporting techniques for product platform development (Simpson et al. 2006). In this section modularization is addressed by showing how several methods and tools are used together for automated derivation of modules assuming that enough architecture-level information is available.

Modularity and decomposition go hand in hand during product development. In order to create appropriate modules, maintaining an overview of the interfaces plays a critical role. These overviews are created in the process of dividing a design task (i.e., decomposing) (Muller 2007; Tomiyama et al. 2007) as part of the effort to manage complexity in the product and its design process (Pahl and Beitz 1996; Suh 1990; Umeda et al. 1990).

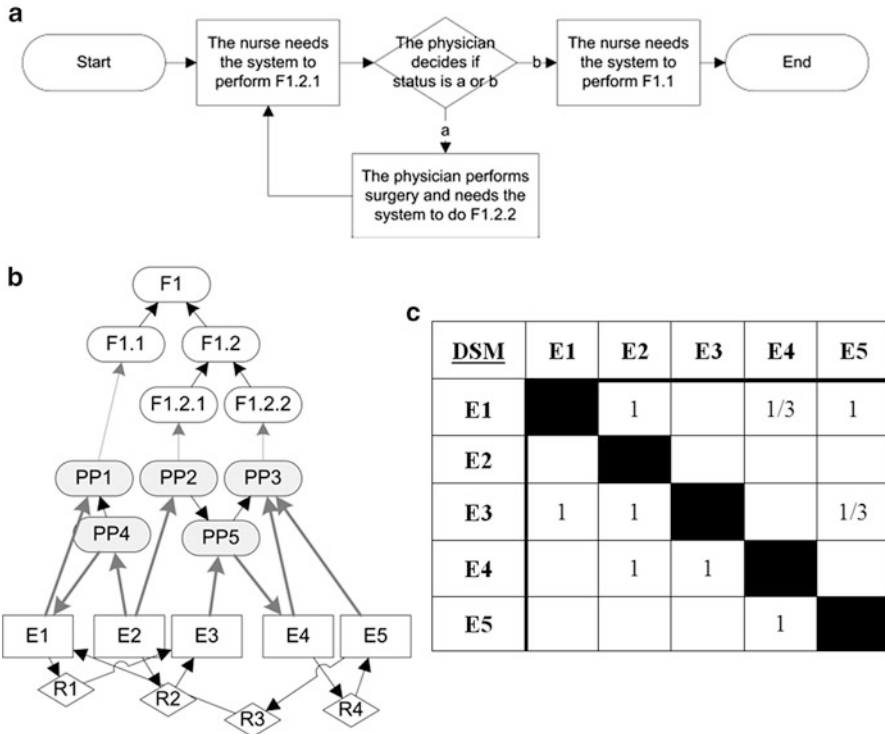
However, in industry, creating and maintaining an overview are not given priority, and the decomposition process is not formally documented during product development. This behavior relates to the perception that the effort investment required by such activities will not be fully paid off later; perception which may be true as long as the current lack of support tools in industry for such development activities persists.

The models presented in this section are taken from the authors' previous work (van Beek et al. 2010; Van Beek and Tomiyama 2010). In those articles, the models and how they are deduced are explained in more detail.

### 17.3.1 Workflow and Function Modeling

To formally document modularization it is necessary to have information on functions, behavior, entities, and their relations to capture the reasoning behind modularization. Function modeling by nature is very suitable to document the top-down path towards modularization. Function modeling can be used to represent the problem decomposition process ("What to do?"). Functions are connected to phenomena (i.e., behavior). Phenomena and static relational descriptions connect entities ("How to do it?").

FBS allows capturing precisely that information but in practice does not seem amenable to the user. Therefore, capturing information is taken to the level of workflows as illustrated in Fig. 17.9a. Thus, this section approaches the



**Fig. 17.9** (a) Simplified workflow model example. *Rectangular nodes* are tasks; *diamond shapes* are decisions; and the *rounded rectangles* denote the start and end of the workflow. The task descriptions point at system functions. (b) Example of FBS model. *F* function, *PP* physical phenomenon, *E* entity, *R* relation. (c) The corresponding filled DSM is shown. The entries are weighted by their path length

modularization problem by using a workflow model, i.e., a flowchart-type diagram that explicitly models the intended user workflow of a future product using a simple natural language. The workflow diagram is accessible and describes how the product will be used in a real application setting combined with any other necessary information (such as preparation, actions of the users, supporting devices). The idea of using workflow models to capture functional requirements in this research was inspired by the field of business process modeling (Van der Aalst 1998) (Figs. 17.10 and 17.11).

Workflow models contain much functional information and can be considered as function models in disguise. The flowchart-type description is understandable for all stakeholders, most importantly the end user, without prior training. In the case of medical systems, for example, the end users would be physicians and nurses. (System architects, engineers, and designers are not allowed to operate the medical systems for patients, so they cannot obtain real use experiences.) Early validation of these workflow models ensures consistent customer expectation and interpretation of the design problem by the system architect. Next, the system architect develops the workflow models into functional system decomposition and a complete FBS model.

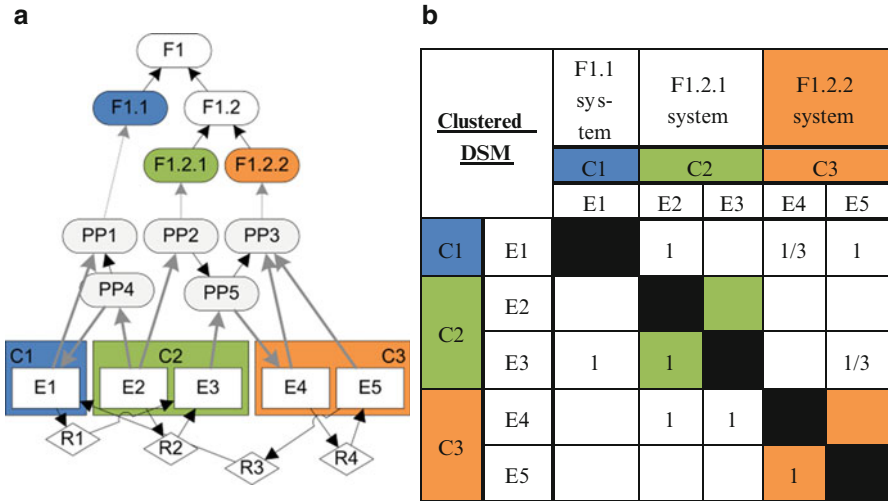


Fig. 17.10 (a) Shows the clustered FBS model. (b) The top row of the DSM shows the names determined by the automatic naming algorithm. The corresponding nodes are filled with corresponding colors in the FBS model on the left. C = cluster

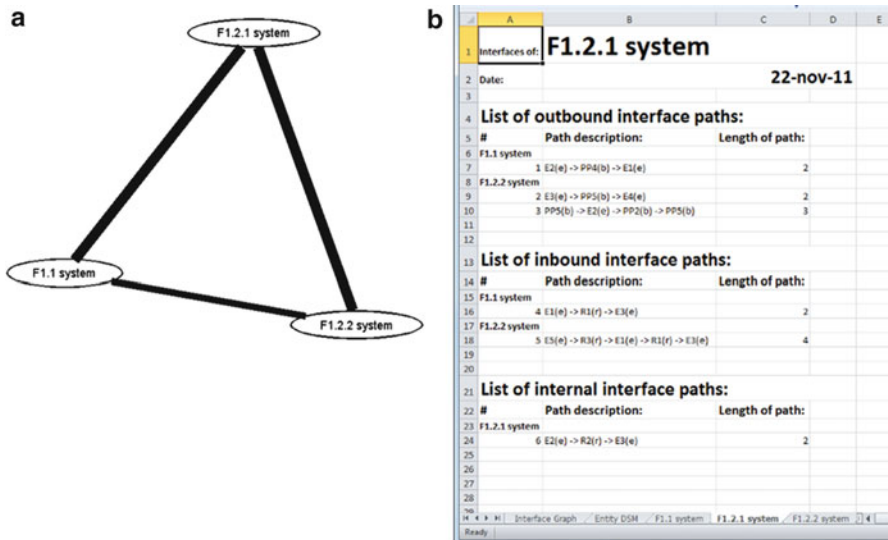


Fig. 17.11 (a) Automatically generated interface overview graph. (b) Automatically generated interface spreadsheet for system “F1.2.1” (the automatically assigned name based on functional connectivity)

Once an FBS model is available (Fig. 17.9b), all entities required for modularization are known (Fig. 17.9c). Modularization is applied to cluster strongly connected entities into modules. For small systems, it might be possible

to perform this task manually. For larger, complex systems this will be a very tedious and error-prone process.

Design Structure Matrix (DSM)-based modularization with clustering techniques (e.g.,  $k$ -means clustering) has been proposed by our group and other researchers (Baldwin and Clark 2006; Browning 2001; Danilovic and Browning 2007; Fernandez 1996; Pimpler and Eppinger 1994; van Beek et al. 2010; Whitfield et al. 2002). The DSM is a matrix representation with a row and column for every system entity. Off-diagonal entries in the DSM represent an interface between the entities of the corresponding row and column. Mathematical clustering algorithms rearrange the rows and columns such that strongly related entities are grouped together into modules.

### ***17.3.2 Modularization and Interfaces***

The DSM is a computer- and mathematics-oriented system representation but not aimed at manual operations. Consider a system with one hundred entities. The DSM would have ten thousand possible entries to go over. Filling the DSM entries manually can be an error-prone and tedious task. Therefore filling the DSM was automated by reusing the FBS model (van Beek et al. 2010) ensuring consistency of high level of abstraction functional models with modularization results. Additionally, the subjective design task is explicitly positioned at the functional level of the system architecting process and becomes more graphic for the architect.

The algorithm determines the number of paths from every function in the functional model to every module. This gives a degree of membership of each module to each function. Mathematically this technique comes down to DSM by Domain Mapping Matrices (DMM) (Danilovic and Browning 2007) multiplications. Proceeding from the top-down, all the modules “pick” a function. When multiple modules pick the same function, iteration is performed and the modules have to pick a child function. This continues until all modules pick different functions. This algorithm abandons the need for human intervention in the modularization process. Conventionally human intervention in the naming of modules was necessary to create presentable results understandable for all stakeholders.

### ***17.3.3 Formula Student Case***

As mentioned in the previous subsection, the method and tools presented here are especially advantageous when dealing with complex systems. For the purpose of this book, a balance had to be found in presenting a nontrivial system and managing the size of the figures. A shifting system of the Delft University of Technology formula SAE car of 2010 fits these requirements.

Figure 17.12 presents the FBS model of the shifting system. The function and phenomena nodes are colored according to membership to the different modules.

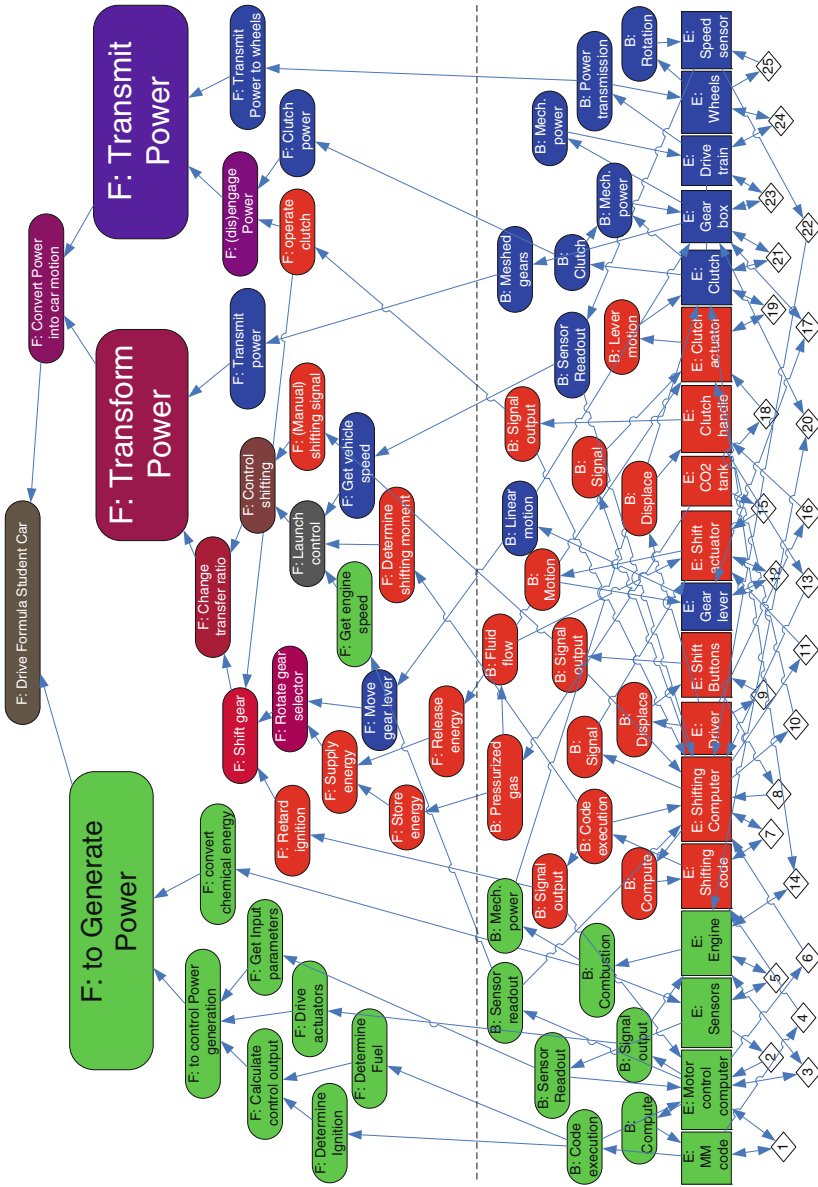


Fig. 17.12 Clustered FBS model of the Formula SAE shifting system according to behavioral dependencies



	Shifting code	Shifting computer	Driver	Shift buttons	Shift actuator	CO2 tank	Clutch handle	Clutch actuator	Gear lever	Clutch	Gear box	Drive train	Wheels	Speed sensor	MM code	Motor Man. computer	Sensors	Engine	
	5	6	7	8	10	11	12	13	9	14	15	16	17	18	1	2	3	4	
Shifting code	5	1	0.6		0.2			0.2							0.2		0.2		
Shifting computer	6	0.6	1	0.4	0.6		0.6							0.2	0.6	0.2	0.6	0.2	
Driver	7			1															
Shift buttons	8			0.6	1														
Shift actuator	10	0.2	0.6		0.2	1	0.4	0.2						0.2		0.2			
CO2 tank	11					1													
Clutch handle	12			0.6				1											
Clutch actuator	13	0.2	0.6		0.2		0.4	0.2	1					0.2		0.2			
Gear lever	9		0.2			0.6			1										
Clutch	14		0.2					0.6		1									0.4
Gear box	15				0.2			0.2	0.6	1	1								
Drive train	16								0.2	0.2	0.6	1							
Wheels	17										0.2	0.6	1						
Speed sensor	18											0.2	0.6	1					
MM code	1														1	0.6	0.2		
Motor Man. computer	2	0.4													0.6	1	0.6	0.2	
Sensors	3															0.2	1	0.6	
Engine	4														0.2	0.6	0.2	1	

Fig. 17.13 DSM generated using FBS model

Based on the Module Strength Indicator (MSI) (Whitfield et al. 2002), the modularization algorithm (van Beek et al. 2010) determined that a decomposition into three modules (Fig. 17.13) minimizes the number of interfaces between modules and maximizes the interfaces inside modules. Figure 17.13 shows the deduced DSM which was used as the input for the modularization algorithm. The modules are colored corresponding to Fig. 17.12.

In the spreadsheet shown in Fig. 17.14, the detailed description about the interfaces in the shifting system is presented, while Fig. 17.15 shows an overview of the interfaces with graphs. The first graph (Fig. 17.15a) shows the interfaces in the shifting system when no modularization is applied. Although the shifting system consists of just eighteen entities, many interfaces can exist. Thicker interface lines denote a stronger relation between two entities.

In Fig. 17.15b the modularization is applied to the overview. Three modules are depicted and named using the automatic naming algorithm. The names are: generate power system, transmit power system, and transform power system. The “generate power system” module, for example, contains the engine, sensors, motor management computer, and motor management computer code. Naming these entities as the “generate power system” seems to fit quite well.

From Fig. 17.15b two stronger interfaces can be denoted. The first strong interface occurs between the power generation system and the power transform system. The second strong interface connects the transform power system to the



	A	B	C	D
1	Interfaces of:	<b>transform power system</b>		
2	Date:	23-nov-11		
3				
4	<b>List of outbound interface paths:</b>			
5	<b>#</b>	<b>Path description:</b>	<b>Length of path:</b>	
6	<b>transmit power system</b>			
7	1	shift actuator(e) -> motion(b) -> Gear lever(e)	2	
8	2	Clutch actuator(e) -> lever motion(b) -> Clutch(e)	2	
9	3	shift actuator(e) -> r 12(r) -> Gear lever(e)	2	
10	4	Clutch actuator(e) -> r 19(r) -> Clutch(e)	2	
11	5	Shifting computer(e) -> signal(b) -> shift actuator(e) -> motion(b) -> Gear lever(e)	4	
12	6	shift actuator(e) -> motion(b) -> Gear lever(e) -> linear motion(b) -> Gear box(e)	4	
13	7	Shifting computer(e) -> signal(b) -> Clutch actuator(e) -> lever motion(b) -> Clutch(e)	4	
14	8	Clutch actuator(e) -> lever motion(b) -> Clutch(e) -> mech. Power(b) -> Gear box(e)	4	
15	<b>Generate power system</b>			
16	9	Shifting code(e) -> code execution(b) -> signal output(b) -> Motor Man. computer(e)	3	
17				
18				
19	<b>List of inbound interface paths:</b>			
20	<b>#</b>	<b>Path description:</b>	<b>Length of path:</b>	
21	<b>transmit power system</b>			
22	10	Speed sensor(e) -> sensor readout(b) -> Shifting computer(e)	2	
23	11	Gear lever(e) -> r 12(r) -> shift actuator(e)	2	
24	12	Speed sensor(e) -> r 22(r) -> Shifting computer(e)	2	
25	13	Clutch(e) -> r 19(r) -> Clutch actuator(e)	2	
26	14	Speed sensor(e) -> sensor readout(b) -> Shifting computer(e) -> compute(b) -> Shifting code(e)	4	
27	15	Speed sensor(e) -> sensor readout(b) -> Shifting computer(e) -> signal(b) -> shift actuator(e)	4	
28	16	Speed sensor(e) -> sensor readout(b) -> Shifting computer(e) -> signal(b) -> Clutch actuator(e)	4	
29	17	Wheels(e) -> rotation(b) -> Speed sensor(e) -> sensor readout(b) -> Shifting computer(e)	4	
30	<b>Generate power system</b>			
31	18	Motor Man. computer(e) -> sensor readout(b) -> Shifting computer(e)	2	
32	19	MM code(e) -> r 4(r) -> Shifting code(e)	2	
33	20	Motor Man. computer(e) -> r 6(r) -> Shifting computer(e)	2	
34	21	MM code(e) -> code execution(b) -> Motor Man. computer(e) -> sensor readout(b) -> Shifting computer(e)	4	
35	22	Sensors(e) -> sensor readout(b) -> Motor Man. computer(e) -> sensor readout(b) -> Shifting computer(e)	4	
36	23	Motor Man. computer(e) -> sensor readout(b) -> Shifting computer(e) -> compute(b) -> Shifting code(e)	4	
37	24	Motor Man. computer(e) -> sensor readout(b) -> Shifting computer(e) -> signal(b) -> shift actuator(e)	4	
38	25	Motor Man. computer(e) -> sensor readout(b) -> Shifting computer(e) -> signal(b) -> Clutch actuator(e)	4	
39				
40				
41	<b>List of internal interface paths:</b>			

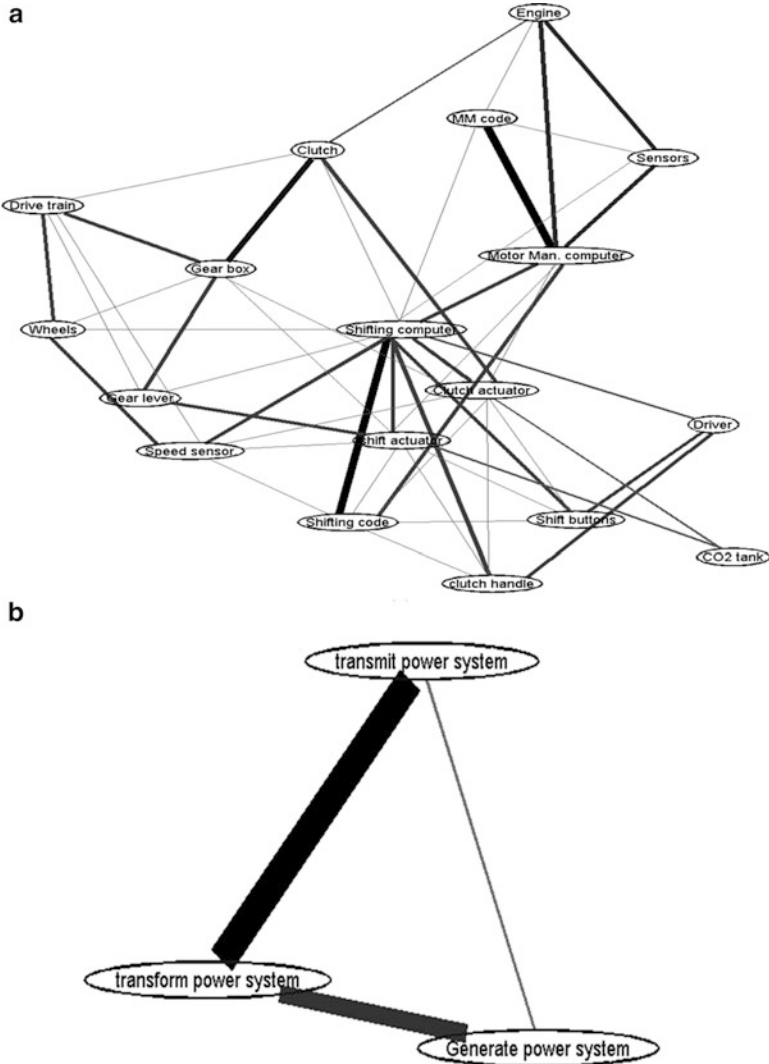
Fig. 17.14 Part of interface specification example for the formula SAE shifting system. This sheet represents the interface specification for the “transform power” system

transmit power system. This resembles the physical setup of the car’s shifting system.

The interface graphs in Fig. 17.15 are suitable for overview, but when the system architect or other stakeholders want to go into detail, the interface spreadsheet in Fig. 17.14 is more appropriate. In that spreadsheet style presentation, the stakeholders can examine what each interface line in the graphs consists of.

## 17.4 System Architecture Models and Their Use in Model-Based Platform Development

This section starts by introducing one standard definition of architectural model (or description) to explain how such a model can help obtaining an overview of design and support model-based development, conforming what is called here architecture-centric development. The section ends with examples demonstrating



**Fig. 17.15** Generated interface graphs model of the shifting system. In (a) the un-modularized graph shows all the entities from the FBS diagram. In (b) the interface graph of the modularized system. The naming of the nodes in (b) was determined automatically using functional connectivity information from the FBS model

how a proposed modeling language and tool have been used to obtain the previous goals in a case study. The aforementioned points can be visualized (cf. Fig. 17.16) as entities (boxes marked with “E”) relating to each other (labeled arrow paths) to accomplish the mapped (dashed lines) functionality (ellipses marked with “F”). Here, a function can be understood as an abstract goal to be achieved. This section mainly covers the work presented in (Alvarez Cabrera 2011).

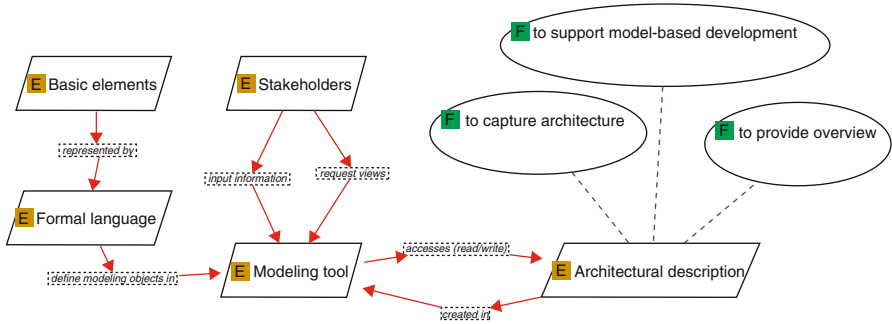


Fig. 17.16 Overview of this section depicted using entities “E” and functions “F”

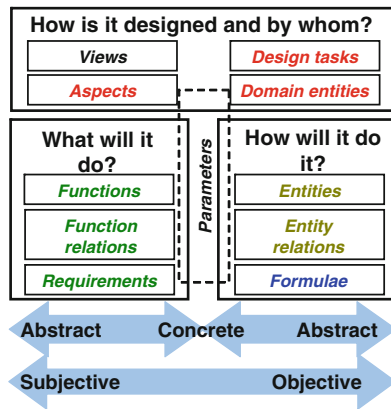


Fig. 17.17 Information spectrum and design questions to be covered by architectural descriptions and the relation to objects proposed in the AM (Alvarez Cabrera 2011)

### 17.4.1 System Architecture and Architectural Descriptions

The definition of system architecture model presented in the introduction provides a good overview of how an architectural description must be and what it should achieve. The basic “elements” and the functionality of a framework enabling the use of a model of the architecture are depicted in Fig. 17.16. However, more precise descriptions are necessary to arrive to a formal language suitable for implementing tooling.

The architectural description must allow representing and integrating information that, due to its varied origin and audience, covers a wide spectrum, ranging between the objective and subjective as well as between the abstract and the concrete. Also it must be possible to declare different types of information to answer the generic and common questions during design (see Fig. 17.17). Under such considerations, the authors have proposed the Architecture Modeling (AM)

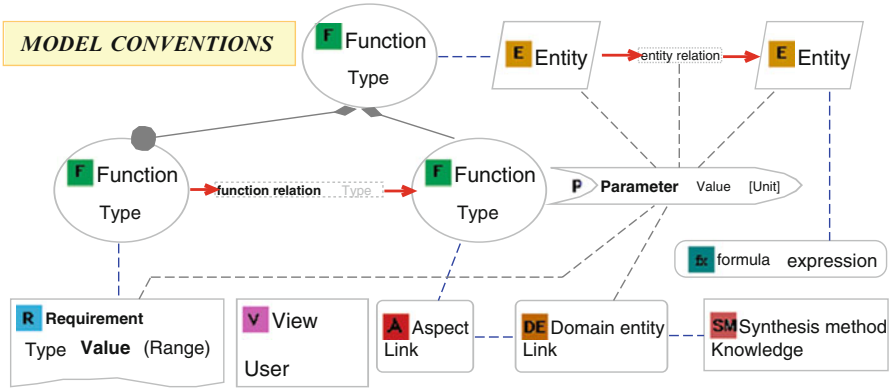


Fig. 17.18 Modeling conventions for objects and relations in the AM (Alvarez Cabrera 2011). Composition connectors (*diamond ends*) aggregate elements of the same class, mappings (*dashed lines*) relate elements of different classes, while relations (*arrow ends*) do it for elements of the same class

language (Alvarez Cabrera 2011) as a base for proper architectural descriptions. More detailed descriptions of this proposal fall out of the scope of this text, and the reader is referred to Alvarez Cabrera (2011), but the basic conventions are provided in Fig. 17.18 to simplify understanding future examples. Note that the same conventions have been used in the description presented in Fig. 17.16.

### 17.4.2 Providing Overview of Development Activities

Returning to the standard presented in Fig. 17.3 and looking beyond the encircled area allow recognizing that the architecture description (and not the architecture) relates strongly to the stakeholders who represent their concerns according to their specific viewpoints. Additionally, the models and views interrelate (not explicitly shown in Fig. 17.3), forming together the architectural description.

The models are not simply linked to an architectural description but also connected to each other through the architectural description in order to represent crosscutting concerns (that can often be represented by so-called key performance indicators).

Such representations of crosscutting concerns (i.e., the views or key performance indicators) help a stakeholder understand the product under development by coupling the overview of the system with the part being developed by this particular stakeholder. However, representing such crosscutting views in a domain-specific language dedicated only to certain stakeholders is not practical considering that the goal is to share part of this view with other stakeholders in order to create a common understanding.

Therefore, defining a common language and formalization play important roles in the proposals of this section. The AM (cf. Fig. 17.18) is one example of such a formal language. Please notice that the specification and communication groups (respectively identified by green and red text in Fig. 17.17) constitute the extension that allows representing the complete set of elements considered here as part of the system architecture (see Fig. 17.1). This language to build an architectural model of a complex multidisciplinary system should support efficiently building a model and views, querying about the model and views, and maintaining consistency of the views (see “request views” relation and “modeling tool” entity in Fig. 17.16). To do so, a special supporting environment called the AM tool was developed (Alvarez Cabrera 2011).

### ***17.4.3 Supporting Information Reuse Through the Model-Based Paradigm***

The AM tool introduced above does more than facilitating view creation; it stores AM instances in a digital format for easy access of other software applications. In this way, the model becomes available for reuse by the users through both manual and automatic means. This section presents examples in which an architectural description is used as a starting point for analysis and reuse while keeping an overview of the development process. The examples go about the industrial development process of part of a printer. The part corresponds to a paper transportation line in the printer known as the paper path. The traditional development process was first documented (using the AM) to increase understanding of the situation.

As shown in Fig. 17.19, an “aspect” (nodes marked with an “A”) makes reference to a common interest of a group of stakeholders and maps to the set of models (nodes marked with “DE”) used to describe and verify properties in the context of such an interest. The aspect handled in this scenario is the paper flow through the engine. This aspect points to domain-specific information on the customer functions (single/double-sided printing), customer requirements (throughput, paper sizes), geometry (component topology), electrical components (motor profiles), simulation (real-time behavior), and software (the timing algorithms and embedded software). The main functionality provided by the paper path is handling the sheets to be printed. Subfunctions include storing, feeding, positioning, heating, and transferring sheets. The input for the timing performance analysis tool, a MATLAB application developed in-house and named Happy Flow (Beckers et al. 2007), contains data on the geometric distribution of the paper path, the type and position of sensors and pinches, and the various operation modes, defined in terms of path segments through which the sheet should be moving. Currently, this data is obtained from various disciplines (cf. Fig. 17.19): a dedicated two-dimensional drawing is transformed into an input file by manually indicating which parts of the drawing belong to the paper path, the location of

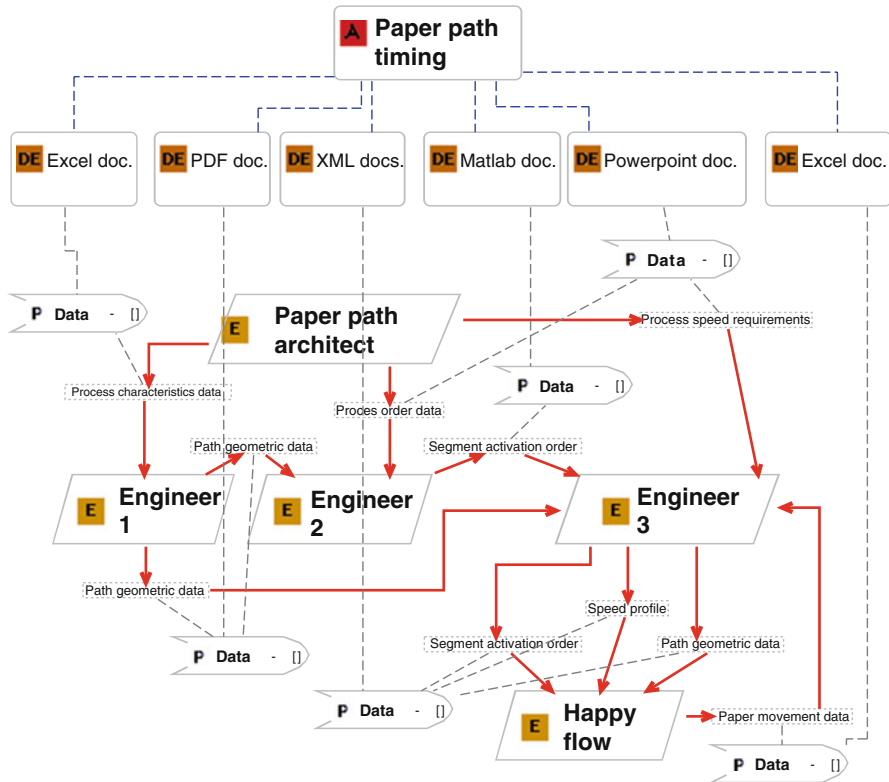
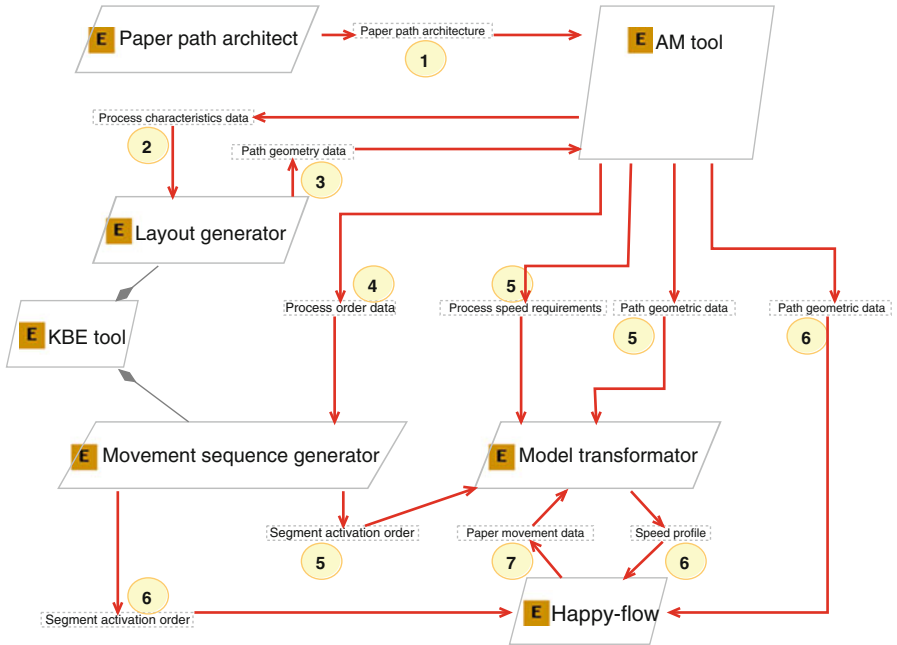


Fig. 17.19 Exchange of information in traditional development process for a paper path

sensors and pinches (pairs of roller to transport papers), and the start and end point of the path segments. Based on the latter information, the operation modes can subsequently be defined. This is a time consuming task that only involves data gathering and deterministic transformations. Furthermore, these steps have to be performed after every noticeable design change.

The domain-specific models that follow the sequence described in Fig. 17.19 are scattered. At each step, the models are made by specialists (using their own languages and tools) based on constraints from a text document (Microsoft Word, Visio, or Excel mostly) supplied by the previous step. After models are verified against the input constraints, the information is summarized by hand in a text document and handed over to the next stage. This approach has a number of disadvantages including lack of ownership, expensive and less reliable manual updating, and lack of overview and context.

With the understanding of the traditional process, an architecture-centric development process was proposed and implemented as shown by the model in Fig. 17.20.

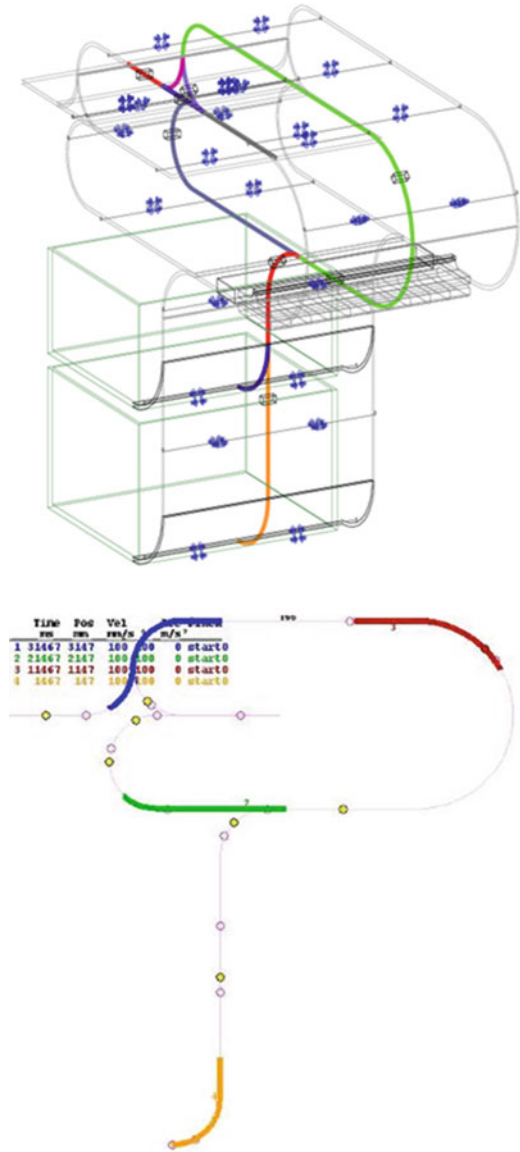


**Fig. 17.20** Model-based development process for a paper path with support from architectural descriptions. Numbers have been added to indicate the design process sequence

This specific AM redefines the process depicted in Fig. 17.19 by specifying the required handover between models in a parametric and reusable way. Each stakeholder provides some part to the aspect model, and the information can be accessed on request without manual transfer. When a shared parameter value is changed, any engineer or tool can update their domain-specific models to the new value, keeping the whole aspect concurrently verified. For the discussed example, a knowledge-based tool (taking the role of the domain-specific tool) was developed to execute the following steps:

- Generate a conceptual geometry of the paper path (wireframe in Fig. 17.21, top), based on the component specification in the AM and a restricted set of design and engineering rules and high-level design parameters. This assumes that the system is composed of existing components and that the set of design rules is complete.
- Determines the number and position of components of the paper path (small markers in Fig. 17.21, top), depending on the current geometry and taking into account design constraints.
- Partitions the paper path in segments (thick-colored lines in Fig. 17.21, top), following the specific segment definition for the Happy Flow simulation tool.
- Determines segment ordering for various operation modes, which are defined using function sequences and component mappings in the AM; see Fig. 17.22.

**Fig. 17.21** Generated geometry with segments (*top*) and snapshot of resulting animation (*bottom*) for the operation mode defined in Fig. 17.22



- Exports component objects (sensors, pinches, etc.), domain-specific objects (segments), and parameters (segment order, shape definition, pinch and sensor positions) to the AM, containing all necessary data to generate the Happy Flow input file, which in turn can perform paper movement simulations (see Fig. 17.21, bottom).



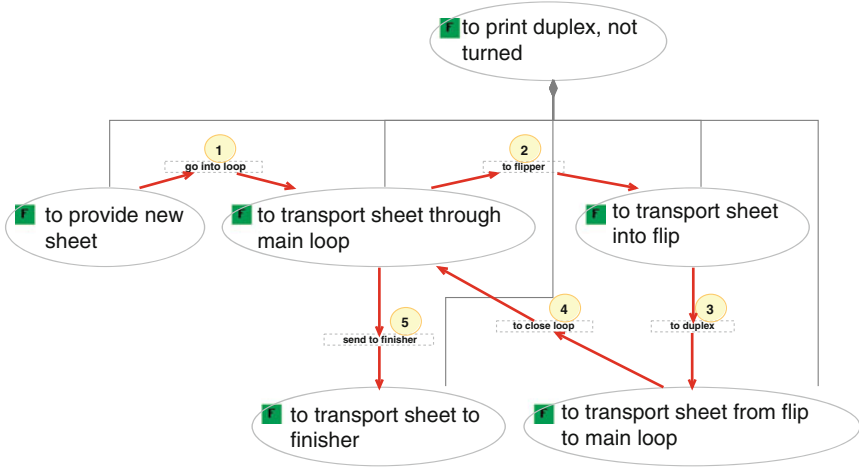


Fig. 17.22 Definition of an operation mode in terms of function objects in the AM. Numbers have been added to roughly indicate the printing process sequence

### 17.5 Conclusions

The existing literature and industry experience indicate that analysis and modeling at the architecture level is necessary for platform development, just as it is necessary for complex product development. Such necessity comes from the fact that the platforming activities revolve around managing commonality, and to achieve this efficiently the stakeholders must maintain an overview of the product and its related processes which simultaneously are linked to the detailed information. Difficulties to achieve this have more than one cause, but the inherent multidisciplinary rising in modern product development can be identified as one of the main root causes of such difficulties. The proposals in this chapter aim mostly to solve such problems coming from multidisciplinary.

With a view to supporting the platforming activities, this chapter has presented prototype tools that allow maintaining an overview while at the same time permit and facilitate access to detailed information for the purpose of analysis and evaluation. Formalization of architecture-level information has been indispensable for the support.

As evidenced by industrial practices, proper support tools for architecting are still not in an acceptable stage of development. Without such tools, supporting the platforming activities is also hard to achieve. Nonetheless, this chapter has provided several ideas which can be used as a backbone and serve as building blocks for such tooling. These ideas include the use of architecture-level information as a backbone as well as prototype tools and techniques to model and use such on formation.

Architecting is deeply related to decomposition and aggregation tasks. Such tasks are commonly performed with difficulties to maintain an overview and scarce modeling support. The first two examples in this chapter present how such tasks can be performed by analyzing either behavioral information at the parametric level or at the structural relation level to form clusters. It has also been shown how the formed clusters can be automatically assigned to meaningful information at the functional level, facilitating the interpretation of the results.

The third example in this chapter has shown how a system architecture model can be used to represent both the product and its relation to the development processes. The model has also shown that this perspective is fundamental if information from several domain-specific models is to be used to evaluate architectures.

**Acknowledgement** This article is based on the research work performed at Delft University of Technology, the Netherlands, between 2006 and 2011 by the authors' group. The authors gratefully acknowledge the following projects that supported the part of the research work: "Automatic Generation of Control Software for Mechatronics Systems" (supported by the Innovation-Oriented Research Programmed "Integral Product Creation and Realization (IOP IPCR)" of the Dutch Ministry of Economic Affairs, Agriculture and Innovation) and "Darwin" and "Octopus" (carried out under the responsibility of the Embedded Systems Institute in Eindhoven and partially supported by the same ministry under the BSIK program). Philips Healthcare and Océ Technologies were industrial partners of the Darwin and Octopus projects, respectively.

## References

- Alvarez Cabrera AA (2011) Architecture-centric design: modeling and applications to control architecture generation. Dissertation, Delft University of Technology
- Avigad G, Moshaiov A, Brauner N (2003) Towards a general tool for mechatronic design. In: Proceedings of 2003 I.E. conference on control applications, 23–25 June 2003. pp 1035–1040. doi:[10.1109/cca.2003.1223153](https://doi.org/10.1109/cca.2003.1223153)
- Baldwin C, Clark K (2006) Modularity in the design of complex engineering systems. In: Braha D, Minai AA, Bar-Yam Y (eds) Understanding complex systems, vol 14. Springer, Berlin, pp 175–205. doi:[10.1007/3-540-32834-3\\_9](https://doi.org/10.1007/3-540-32834-3_9)
- Beckers J, Heemels W, Bukkems B, Muller G (2007) Effective industrial modeling for high-tech systems: the example of happy flow. In: 17th annual symposium of INCOSE, San Diego, CA, USA, 24–28 Jun 2007. INCOSE, pp 1–12
- Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Trans Eng Manage* 48(3):292–306
- Craig K, DeVito M, Mattice M, LaVigna C, Teolis C (1999) Mechatronic integration modeling. In: Proceedings of IEEE/ASME international conference on advanced intelligent mechatronics, 1999. pp 1032–1037. doi:[10.1109/aim.1999.803314](https://doi.org/10.1109/aim.1999.803314)
- Danilovic M, Browning TR (2007) Managing complex product development projects with design structure matrices and domain mapping matrices. *Int J Proj Manage* 25(3):300–314. doi:[10.1016/j.ijproman.2006.11.003](https://doi.org/10.1016/j.ijproman.2006.11.003)
- Erden MS, Komoto H, van Beek TJ, D'Amelio V, Echavarria E, Tomiyama T (2008) A review of function modeling: approaches and applications. *Artif Intell Eng Des Anal Manuf* 22(2):147–169
- Erixon G, von Yxkull A, Arnström A (1996) Modularity—the basis for product and factory reengineering. *Ann CIRP* 45(1):1–6. doi:[10.1016/s0007-8506\(07\)63005-4](https://doi.org/10.1016/s0007-8506(07)63005-4)

- Fernandez CIG (1996) Integration analysis of product architecture to support effective team co-location. M.Sc. Thesis, Massachusetts Institute of Technology
- Forsberg K, Mooz H (1991) The relationship of system engineering to the project cycle. In: National Council on Systems Engineering (NCOSE) and American Society for Engineering Management (ASEM), Chattanooga, TN, 21–23 Oct 1991. pp 1–12
- IEEE (2000) Systems engineering and software engineering—Recommended Practice for Architectural Description of Software-Intensive Systems. ISO/IEC 42010—IEEE-Std-1471-2000
- Komoto H, Tomiyama T (2010) A system architecting tool for mechatronic systems design. *Ann CIRP* 59(1):171–174. doi:[10.1016/j.cirp.2010.03.104](https://doi.org/10.1016/j.cirp.2010.03.104)
- Komoto H, Tomiyama T (2011) Multi-disciplinary system decomposition of complex mechatronics systems. *Ann CIRP* 60(1):191–194. doi:[10.1016/j.cirp.2011.03.102](https://doi.org/10.1016/j.cirp.2011.03.102)
- Martinez MT, Favrel J, Ghodous P (2000) Product family manufacturing plan generation and classification. *Conc Eng* 8(1):12–23. doi:[10.1177/1063293x0000800102](https://doi.org/10.1177/1063293x0000800102)
- Muller G (2007) A multidisciplinary research approach, illustrated by the Boderc project. <http://www.gaudisite.nl>
- Pahl G, Beitz W (1996) Engineering design: a systematic approach, 3rd edn. Springer, Berlin
- Pimmler TU, Eppinger SD (1994) Integration analysis of product decompositions. Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA
- Simpson TW, Siddique Z, Jiao J (2006) Product platform and product family design: methods and applications. Springer, Berlin
- Suh N (1990) The principles of design. Oxford University Press, New York, NY
- Tomiyama T, D'Amelio V, Urbanic J, ElMaraghy W (2007) Complexity of multi-disciplinary design. *Ann CIRP* 56(1):185–188
- Umeda Y, Takeda H, Tomiyama T, Yoshikawa H (1990) Function, behaviour and structure. In: Gero JS (ed) Proceedings of the 5th international conference: application of artificial intelligence in engineering, Boston, USA, 1990. Computational Mechanics Publications, Southampton and Springer, Berlin, pp 177–194
- Umeda Y, Ishii M, Yoshioka M, Shimomura Y, Tomiyama T (1996) Supporting conceptual design based on the function-behavior-state modeler. *Artif Intell Eng Des Anal Manuf* 10(4):275–288
- Van Amerongen J (2003) Mechatronic design. *Mechatronics* 13(10):1045–1066
- Van Beek TJ, Tomiyama T (2010) Combining user workflow and system functions in product development. In: ASME conference proceedings, Montréal, Canada, 2010. pp 239–248
- Van Beek TJ, Erden MS, Tomiyama T (2010) Modular design of mechatronic systems with function modeling. *Mechatronics* 20(8):850–863. doi:[10.1016/j.mechatronics.2010.02.002](https://doi.org/10.1016/j.mechatronics.2010.02.002)
- Van der Aalst W (1998) The application of Petri nets to workflow management. *J Circuit Syst Comp* 8:21–66
- Whitfield R, Smith J, Duffy A (2002) Identifying component modules. In: 7th International conference on artificial intelligence in design, Cambridge, 7 Dec 2002. pp 571–592
- Xu Y, Zou H (2007) Design principles for mechatronic systems based on information content. *Proc Inst Mech Eng B J Eng Manuf* 221(7):1245–1254

# Chapter 18

## Product Family Commonality Selection Using Optimization and Interactive Visualization

Ritesh Khire, Jiachuan Wang, Trevor Bailey, Yao Lin,  
and Timothy W. Simpson

**Abstract** High dimensionality and computational complexity are curses typically associated with many product family design problems. In this chapter, we discuss interactive methods that combine two traditional technologies—optimization and visualization—to create new and powerful strategies to expedite high dimensional design space exploration and product family commonality selection. In particular, three different methods are compared and contrasted: (1) exhaustive search with visualization, (2) individual product optimization with visualization, and (3) product family optimization with visualization. Among these three, the individual product optimization with visualization method appears to be the most suitable one for engineering designers who do not have a strong optimization background. This method allows designers to “shop” for the best designs iteratively, while gaining key insight into the trade-off between commonality and individual product performance. The study is conducted in the context of designing a product family using an in-house, system-level simulation tool. The challenges associated with (1) design space exploration involving mixed-type design variables and infeasibility and (2) visualizing *product family design spaces* during commonality selection are addressed. Our findings indicate a positive impact on the company’s current approach to product family design and commonality selection.

---

An earlier version of this chapter appeared in R. Khire, J. Wang, T. Bailey, Y. Lin, and T. W. Simpson (2008) Product Family Commonality Selection through Interactive Visualization, ASME Design Engineering Technical Conferences – Design Automation Conference, New York, NY, ASME, Paper No. DETC2008/DAC-49335 (© ASME 2008), reprinted with permission.

R. Khire (✉) • J. Wang • T. Bailey • Y. Lin  
United Technologies Research Center, East Hartford, CT 06118, USA  
e-mail: [KhireR@utrc.utc.com](mailto:KhireR@utrc.utc.com)

T.W. Simpson  
Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA  
Industrial and Manufacturing Engineering, Penn State University, University Park,  
PA 16802, USA

## 18.1 Introduction

In many technology-focused companies, engineering practice is evolving to a state where rigorous physics-based models are being used to analyze and verify product design performance and reliability. Increased product complexity and competitive pressure to accelerate product introductions to market have motivated large companies such as Boeing, Ford, Caterpillar, and United Technologies Corporation (UTC) to pursue rigorous analytical approaches to engineering design (Simpson and Martins 2011). At UTC, for instance, benchmarks have shown that a commitment to model-based design, analysis, and verification can provide as high as 30 % savings in engineering costs and can also cut development time in half.

To complicate matters further, most companies now offer many of their products as variants within a larger product family. A product platform, consisting of common components or subsystems across the family, is typically used to generate high profits (Meyer and Lehnerd 1997). A typical product family consists of a set of products that have (1) some unique characteristics and (2) shared components and modules. As such, product family optimization inherits all of the idiosyncrasies involved with single product optimization (e.g., multiple objectives, mixed variables), while adding coordination across the family. The latter involves additional intricacies such as making commonality decisions, significantly high dimensionality, and combined combinatorial-attribute decision making. This creates additional challenges (such as high computing requirements) when physics-based models are used for design and verification of product families. We refer the reader to work by de Weck (2005), which provides a thorough example of the types of models needed to translate product family design decisions into profitability for a company.

Typical steps in product family optimization include (1) defining the product family, (2) formulating the product family optimization problem, (3) solving the product family optimization problem, and (4) evaluating the trade-off between different product family design alternatives and making a final decision (Simpson 2005). In this chapter, we discuss the challenges involved with formulating the product family design problem within an industrial setting. Thereafter, we compare and contrast three methods that focus on the last step of the aforementioned process, i.e., evaluating the trade-off and making a final decision. While the arguments, challenges, methods, and results discussed herein are within the context of a specific problem, we assert that our findings are generalizable to industrial problems of similar complexity and of comparable levels of technical difficulty.

The next section reviews challenges in product family design and optimization and related work in this area. Section 18.3 introduces the product family design example used in this work. Section 18.4 presents, compares, and down-selects between the three methods that we are investigating. Section 18.5 discusses the results from using the down-selected method. Section 18.6 offers closing remarks.

## 18.2 Related Work in Product Family Design

As mentioned earlier, a typical product family consists of a set of products that share common components and/or modules. There are many advantages of sharing common components (Collier 1981), including (1) economies of scale, (2) reduced development time, (3) reduced SKU (Stock Keeping Units), (4) reduced manufacturing and service complexity, and (5) increased product quality due to less part variety. From a business perspective, these advantages can be tied to low-cost products or increased profitability of a product line. However, these advantages must be carefully weighed against the potential disadvantages of commonality.

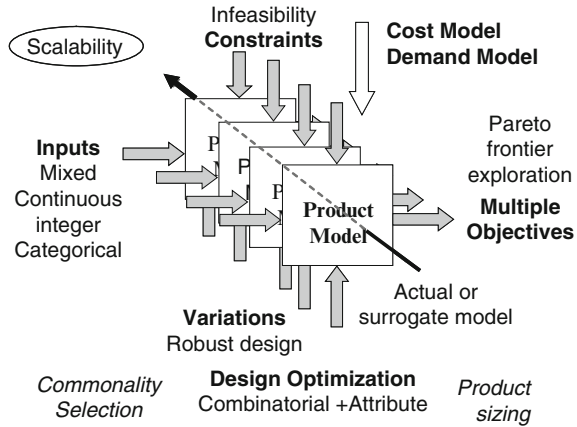
Perhaps the biggest drawback to commonality is the increased potential for the lack of product distinctiveness (Robertson and Ulrich 1998). As more components are shared, it becomes increasingly difficult to differentiate between product variants in the market (Miller 1999). Moreover, individual product performance may degrade significantly due to commonality, resulting in the loss of market share (Lutz 1998). Therefore, a product family designer must carefully balance the trade-off between commonality and individual product differentiation.

More than 40 optimization-based approaches have been proposed to help resolve this trade-off (Simpson 2005). These approaches can be generally categorized into one of three product family design strategies: (1) select the platform first and then optimize the platform and variants, (2) optimize the variants first and then select the platform that causes the minimum performance loss with maximum commonality savings, and (3) simultaneously select the platform while optimizing the platform and the variants. There are advantages and disadvantages to each strategy (Khajavirad et al. 2007). Note that strategy (1) requires *a priori* selection of the platform. Typically, such selection is based on designer's experience and, often, no systematic processes are used during selection. On the contrary, the three methods presented in this chapter are based on strategies (2) and (3), which provide systematic platform selection processes (see Sect. 18.4).

Regardless of which particular strategy is employed, product family design optimization entails (a) exploration of product family design space and (b) ultimately making a decision regarding the appropriate level of commonality in the family, what we refer to as *commonality selection*. With regards to (a), involvement of multiple products significantly increases the dimensionality and complexity of product family design space, even for modest-sized product families (Messac et al. 2002). As shown in Fig. 18.1, design space exploration of a product family involves not just one product model, but all the individual product models within a product family.

Product family optimization consists of both combinatorial optimization (to select the common components and modules that embody the platform) and parametric optimization (to optimize both the platform and non-platform design variables) (Fujita 2005). Working with mixed-type design variables and multiple conflicting design objectives exacerbates the problem further, as does having

**Fig. 18.1** Challenges in product family optimization: design space exploration

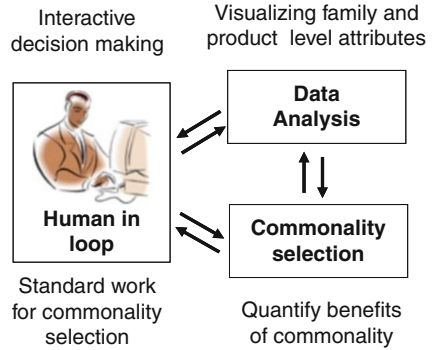


“black box” simulations with implicit constraints that yield many infeasible solutions within the design space. This makes the design space discontinuous such that it is difficult to apply traditional optimization techniques to search the design space. We refer the reader to Chap. 12 for more discussion on this topic.

Once the design space has been explored to the best extent possible, then designers can proceed with (b), namely, determining the appropriate level of commonality for the product family. The challenges associated with commonality selection are the following: (1) it involves the aforementioned trade-off between commonality and individual product performance, which needs human preference guidance, and (2) there are multiple valid solutions for each individual product (Simpson et al. 2012), and commonality selection needs to examine all of the data and make sound judgment (Slingerland et al. 2010).

What we have seen and experienced in practice, however, is that designers, already leery about enforcing commonality (Halman et al. 2005), are all the more hesitant to trust the optimization results, and rightly so given the challenge in formulating an accurate optimization problem that reflects the subjectivity involved in the trade-off between commonality and individual product performance. This reluctance also stems from (1) an innate, albeit unfounded, belief that any commonality will adversely affect the product’s performance combined with (2) the inability to view the trade-offs that are occurring within the design space. It is one thing to visualize how an individual product performs relative to a known baseline design, but it is much more challenging to visualize how an entire family of products performs. Work in this area has been very limited to date (Slingerland et al. 2010), and this motivated our study; namely, to promote commonality, designers must be able to visualize the trade-offs that are occurring in the product family. We propose to take this one step further by putting designers “in the loop”

**Fig. 18.2** Challenges in product family optimization: commonality decision making



during optimization process as shown in Fig. 18.2, leveraging recent research to support “Design by Shopping” approaches to engineering design (Balling 1999; Eddy and Lewis 2002; Stump et al. 2009; Winer and Bloebaum 2001).

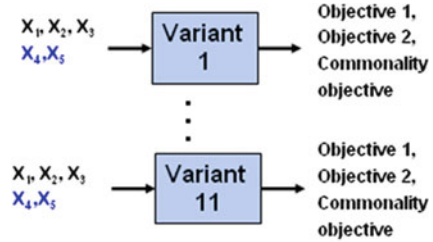
An intuitive approach to make commonality decisions is to present all of the possible solutions to designers and allow them to make commonality decisions interactively. However, this is easier said than done, given the high dimensionality and sheer amount of data associated with a product family. The three methods investigated in Sect. 18.4 provide three different approaches for exploring the design space and then visualizing it before making commonality decisions. Before introducing the methods, however, we first introduce the example product family design problem that has motivated our work.

### 18.3 Product Family Design Example

Figure 18.3 depicts the scope of the product family design example from United Technologies Corporation (UTC) that motivated this work. Because of the proprietary nature of the products being designed, we restrict ourselves to terminologies such as UTC product and UTC product family. The family consists of 11 product variants that are each defined by three continuous and two categorical variables. The overall UTC product family design problem includes a total of 33 continuous and 22 categorical variables, 23 objectives (2 objectives/product plus a commonality index), and more than a hundred constraints. The constraints are internal to the system-level simulation model that was developed in-house to support physics-based design and analysis. The challenges involved in solving the UTC product family design problem are discussed next before being used to illustrate, compare, and contrast the interactive product family design methods in Sect. 18.4.



**Fig. 18.3** Scope of UTC product family design problem



*Variables in blue are categorical*

- 11 Product models
- 33 Continuous variables
- 22 Categorical Variables
- 23 Objectives
- 100+ Constraints

### 18.3.1 UTC Product Design Space and Optimization Algorithm

For the UTC product family problem, we consider five design variables for each product,  $X_1$  to  $X_5$ , of which three are continuous and two are categorical. Unlike continuous variables, the numerical values of categorical variables do not have any physical significance. For example, refrigerant temperature is a continuous variable, as its value indicates hotness or coldness. On the other hand, a compressor model number is a categorical variable, as it does not necessarily contain any physical meaning.

Handling continuous and categorical variables simultaneously poses computational challenges for any optimization environment. Either mixed-integer nonlinear programming (MINLP) or non-gradient-based methods can potentially be used to solve such an optimization problem, and MINLP formulations for product family design are being investigated (Khajavirad and Michalek 2007a). Success of MINLP approaches in finding optimal solutions typically depends on a number of factors, such as starting point, convexity of design space, and continuity and infeasibility associated with the design space. The UTC product family design problem entails a significant level of (1) discontinuity and (2) infeasibility, making it inappropriate for MINLP methods. As such, most of the product family optimization problems are discontinuous in nature due to discrete choices of platform and non-platform variables.

As for the infeasibilities in the UTC product family example, they arise from non-convergence of the system-level simulation model itself—an area of ongoing investigation and future research. Our experience suggests that when the existing UTC product model is randomly sampled within the design space, at best 40 % of the samples yield feasible designs. If we extend this observation to the UTC family of 11 products, the feasibility of the design space would be  $(0.4)^{11} = 0.004 \%$ .

To put that in perspective, the initial sample size of GA algorithm will have to be at least 25,000 to get one feasible solution. This significant infeasibility poses additional challenges to the design space exploration process, impacting the choice of the algorithm used for optimization.

To overcome these discontinuities and infeasibilities, we selected a non-gradient-based evolutionary algorithm for the UTC product family problem. In particular, we use the Non-Dominated Sorting Genetic Algorithm (NSGA-II), a popular multi-objective genetic algorithm (GA) developed by Deb (2001). NSGA-II is robust to discontinuities in the design space and is capable of searching for global optima. An important feature of NSGA-II is its ability to explore Pareto frontiers, and it has been used by many researchers to solve product family optimization problems (Simpson 2005). The objectives for the UTC problem are discussed next.

### ***18.3.2 Objectives for the UTC Product Family***

As seen in Fig. 18.3, each UTC product family involves multiple objectives. Specifically, each product has two objectives, with a preference for maximizing Objective 1 and minimizing Objective 2. This creates 22 objectives for the entire product family, with the 23rd objective being the commonality index for the product family (see Sect. 18.3.3).

Handling multiple instances of similar objectives poses a challenge for product family design. Aggregating similar objectives into a single objective seems to be an intuitive way of tackling the problem at hand. However, aggregation poses some critical computational and practical challenges, such as (1) loss of individuality, (2) handling different scales of objectives (e.g., value of Objective 1 for one of the product variants may be in tens, whereas that for another variant may be in thousands), and (3) handling the relative importance between different units.

Based on our observations, these critical challenges have been handled based on ad hoc rules, which can potentially be a significant source of suboptimality. Decomposition-based optimization strategies can be implemented to alleviate some of these challenges, but at the expense of added computational complexity and coordination cost (Khajavirad et al. 2007). As we see in the next section, visualization-aided decision-making framework appears more effective than simple objective aggregation.

### ***18.3.3 Commonality Selection***

Typically, an engineering system consists of a number of possible commonality choices (components and modules). As such, comprehensive product family design involves exploring potential commonality options for all these components and

modules. In this study, we focus on selecting only one common feature among different UTC products, referred to as the commonality variable.

Quantifying the benefits of commonality is important for making commonality decisions; however, quantifying the benefits of commonality is extremely difficult in practice. Hence, commonality indices are typically used as surrogates to qualify the resulting cost savings. A typical commonality index is a function of the number of components/assemblies/manufacturing processes that are common across different products in the family (Thevenot and Simpson 2006). Khajavirad and Michalek (2007b) argue that the commonality index (CI) introduced by Martin and Ishii (1997) captures the tooling cost savings of component commonality better than any other commonality metric. Using this index as our starting point, our commonality index for this product family optimization problem is expressed as a percent of the number of different commonality variable values used in the family (see Sect. 18.4.3 for more detail). Meanwhile, in Sect. 18.4.2, we show the effectiveness of the visualization-aided commonality selection procedure that does not rely on traditional commonality indices.

## 18.4 Interactive Visualization Methods for Commonality Selection

In this section, we discuss the three methods that we used to solve the UTC product family problem. Table 18.1 summarizes the aspects of each method. We discuss each method in detail in the subsections that follow.

### 18.4.1 Method 1 (Exhaustive + Visualization)

Figure 18.4 shows the steps used in Method 1: exhaustive sampling followed by visualization. In this method, the design space is explored exhaustively (and separately) for each product using a large number of sample points. The number

**Table 18.1** Summary of product family optimization methods evaluated in study

	Design space exploration	Commonality selection	Advantage	Disadvantage
Method 1	DOE sampling	Interactive visualization	Simple, interactive	No optimality, computationally intensive
Method 2	Individual product optimization	Interactive visualization	Interactive, Pareto optimal	Limited variety of product families
Method 3	Product family optimization	Formulated in optimization	Unlimited variety in product families	Computationally intensive, not interactive

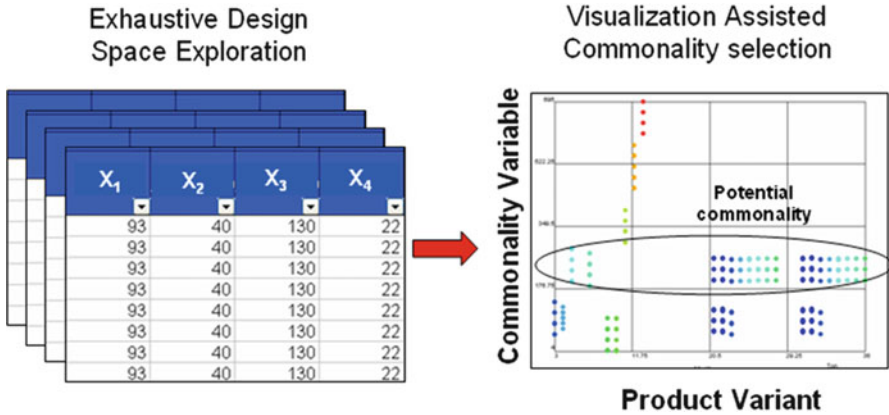


Fig. 18.4 Exhaustive product family design with visualization

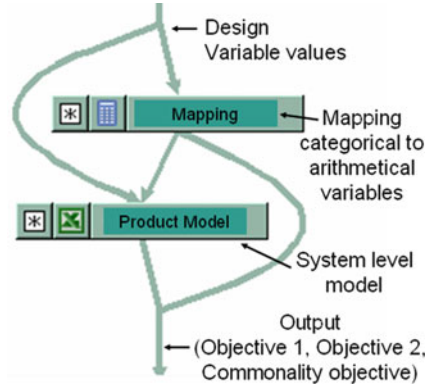
of sample points typically depends on factors such as (1) available time, (2) computational resource, (3) number of design variables, and (4) prior knowledge of the design space. Based on our experience, the first two factors dominated the UTC product family design problem.

After exhaustive sampling, the input and output data for each product is assembled into a single file location (typically an Excel spreadsheet or a text document) for product family design. In particular, the product family data is processed to enable commonality decisions using any of a variety of filtering tools or sorting techniques. For example, when selecting the value of commonality variable, the designer can successively filter different commonality variable values to identify UTC products that share a particular variable value.

According to our experience, such text-based filtering is unfriendly and time consuming for designers. They prefer visual tools to do this processing interactively—such techniques are simpler, more user-friendly, and allow the designer to explore larger design spaces with ease. Also, we found that the quality (trade-off between the objectives) of the design selected using visualization technique is frequently superior to that obtained from the aforementioned text-based filtering techniques. Since the specific visualization techniques are also used in Method 2, they are discussed in the next section after introducing Method 2.

In summary, the advantages of Method 1 are that it is easy to use and it retains many aspects of the engineer's current design practice. While this latter point may seem counterintuitive, it is critical when considering adoption of new techniques in current practice. Disadvantages of this method are that exhaustive sampling may not uniformly sample the objective space and the computational expense of the simulation model may limit the number of samples that can be obtained. Both of these may lead to designs that have poor trade-off between design objectives, creating an opportunity to utilize optimization to search the design space more efficiently as advocated in Method 2 as discussed next.

**Fig. 18.5** Optimization of individual product variants



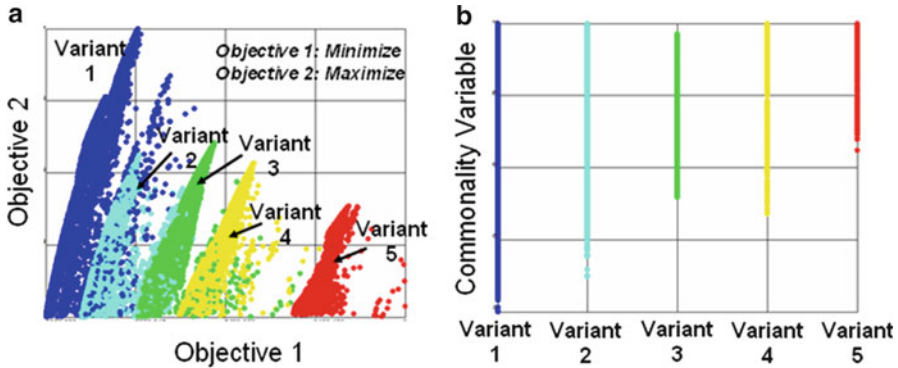
### 18.4.2 Method 2 (Individual Opt + Visualization)

Method 2 is similar to Method 1 except that the exhaustive design space exploration used in Method 1 is replaced by systematic optimization of individual products in Method 2 to search the design space more efficiently. Figure 18.5 shows the integration of the UTC simulation model within Engineeus' iSight environment (Koch et al. 2002) for performing individual optimization. As discussed earlier, the UTC product design problem includes categorical variables, which are mapped to arithmetic variables before sending them to the system-level model, as shown in Fig. 18.5. Next, we discuss the use of visualization for exploring the *product family design space*.

#### 18.4.2.1 Interactive Visualization

After individual optimization of all the products is completed, multidimensional data visualization is used to perform commonality selection. For this work, we employ the Applied Research Laboratory's Trade Space Visualizer (ATSV), a Java-based application that displays multidimensional data using glyph, histogram, scatter, scatter matrix, and parallel coordinate plots (Stump et al. 2009). The ATSV is developed entirely in Java, making it cross-platform compatible, and offers linked views, brushing (filtering), preference shading, and Pareto filtering to help designers "shop" for the best design, which, in this case, is the best product family given a selected level of commonality.

In the past, ATSV has been used primarily for single product optimization. This work extends the use of ATSV to product family optimization. As such, assembling data from individual optimizations of different products into a single file is a part of customizing ATSV for product family optimization. An important outcome of the current research is the identification of product family-specific capabilities for ATSV, such as data preprocessing for aggregating variant data.



**Fig. 18.6** Product family design space obtained from individual optimization. (a) Efficiency vs. cost. (b) Commonality vs. variants

Figure 18.6 shows the design space for five variants in the UTC product family. Specifically, Fig. 18.6a shows Objective 1 versus Objective 2 space for different UTC product variants, while Fig. 18.6b shows available commonality variable values for different variants. Interestingly, Fig. 18.6b suggests that commonality variable can potentially be shared by all five product variants by setting its value high. This shows the simplicity and effectiveness of multidimensional data visualization in exploring the design space for a complete product family. Also, visualization (see Fig. 18.6) has provided high-level information to the engineering designer regarding possible commonality choices in the family. Next, we use the commonality variable data shown in Fig. 18.6b for making commonality selections. The impact of commonality selection is evaluated using the Objective 1 versus Objective 2 data shown in Fig. 18.6a.

#### 18.4.2.2 Commonality Selection with Visualization

Figure 18.7a, c shows the effect of commonality variable values A and B on Objective 1 and Objective 2. Also, Fig. 18.7b shows the brushing controls in ATSV, which act as a sliding data filter to allow users to evaluate different commonality selections.

In Fig. 18.7b, the brushing is set for the commonality variable. As the brush slides from left to right, the designs that have the corresponding commonality variable values are shown in colors. On the other hand, the designs with commonality variable values different from that selected by the brush are automatically turned grey, as seen in Fig. 18.7a, c. Thus, by simply sliding the brush controller, the designer can visually evaluate the effect of change in the commonality variable value on the two objectives for all the products in the family.

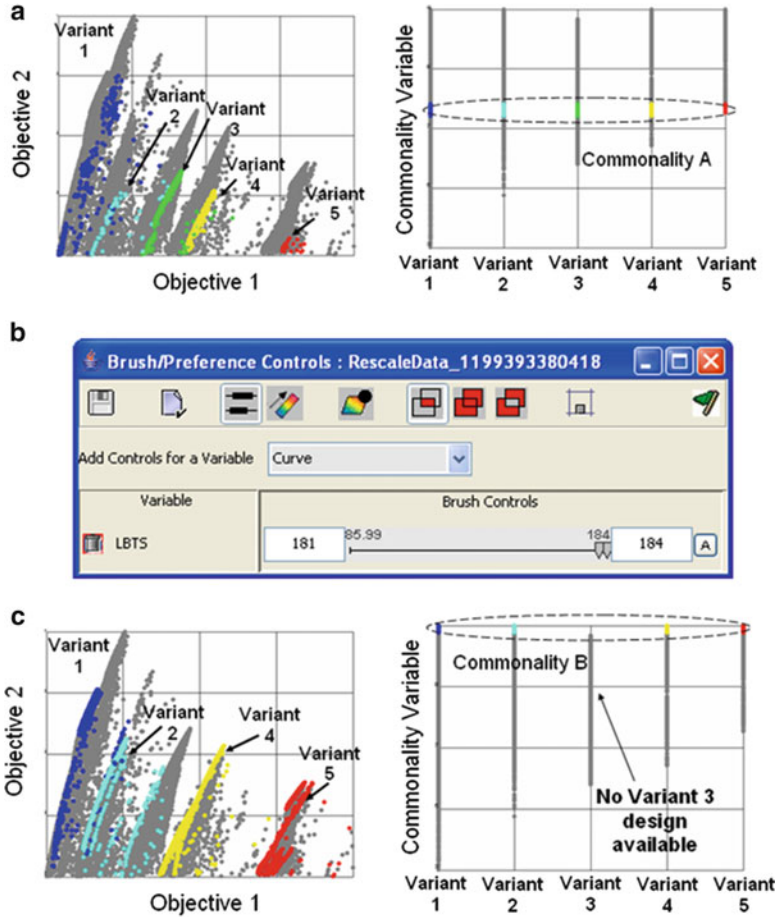


Fig. 18.7 Effect of different commonality selection. (a) Effect of commonality variable values A. (b) Brushing control for commonality selection. (c) Effect of commonality variable values B

We further explain the commonality selection with the help of Fig. 18.7a, c. By setting the brush at Commonality A, we generate Fig. 18.7a. We can observe that commonality variable value A can be made common between all variants of the UTC products units. Additionally, we can also observe that commonality variable value A results in designs that are on the Pareto frontiers of the most UTC products. On the other hand, commonality variable value B does not result in any feasible design for product variant 3. Hence, commonality variable value B cannot be made common for the entire product family.

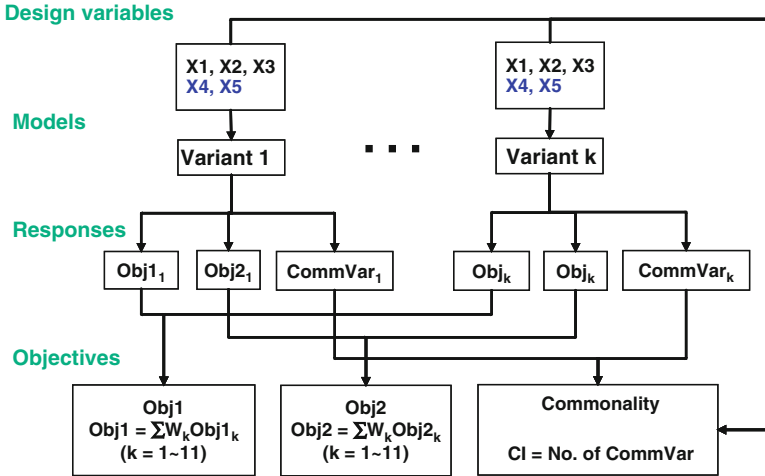


Fig. 18.8 All-in-one product family optimization

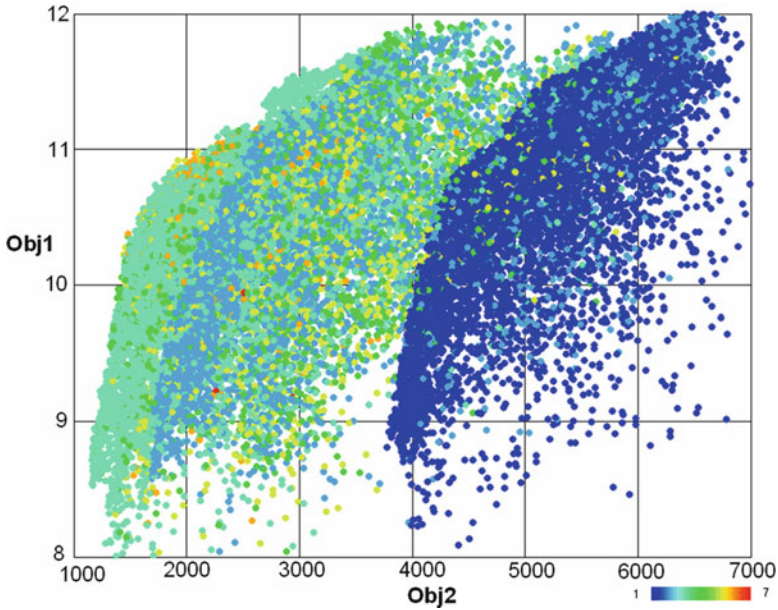
Thus, we observe that by combining individual product optimization with interactive visualization, we have developed a powerful method that is simple yet effective in making commonality selection. Additional advantages of Method 2 include an improved ability to find designs with better trade-off resolution (all objectives show simultaneous improvement) over Method 1, added flexibility while still being moderately easy to use. A disadvantage of Method 2 is the post-processing of the individual product data to aggregate it for visualization. Also, there is a conflict of interest in that the optimization drives towards Pareto optimality for the individual products, while commonality selection may force you away from these individual Pareto frontiers for the benefit of the family. Solutions obtained from the individual optimization may not always be most appropriate for commonality selection, which is why visualization is all the more critical at this stage of the product family design process.

### 18.4.3 Method 3 (Product Family Opt + Visualization)

Within the context of the UTC product family design problem, we have also applied two product family-based optimization techniques: (1) all-in-one approach and (2) decomposition-based approach (Khajavirad et al. 2007). As the name implies, the all-in-one approach takes all the inputs and outputs of each individual product optimization problem and combines them into a large optimization problem. A typical all-in-one optimization problem uses commonality index as an additional objective that ties all the products together.

Formulation of the all-in-one approach for the UTC product family design problem is shown in Fig. 18.8. Typically, an all-in-one problem scales the number



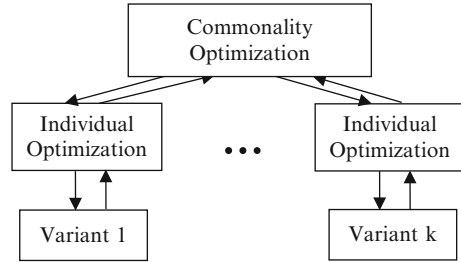


**Fig. 18.9** Results from product family optimization

of design variables, constraints, and objectives based on the number of product variants. This scaling increases the complexity of the all-in-one optimization problem. The two objectives, Obj1 and Obj2, shown in Fig. 18.8 are obtained from aggregating corresponding objectives from each product. The Commonality Index (CI), in this case, is simply the number of distinct values taken by the commonality variable across the product family. The CI ranges from 1 to  $k$ ;  $k$  indicates how many different values of the commonality variables are used across the products, while 1 implies that all the products share the same commonality variable value.

For the all-in-one approach, visualization techniques can also be applied to help designers make commonality decisions. Figure 18.9 shows an illustrative scatter plot (Obj1 vs. Obj2) for a commonality study. We want to maximize Obj1 while at the same time minimize Obj2. Each point on Fig. 18.9 represents a possible solution from the all-in-one optimization. Without a visualization tool, the designers were not able to differentiate between the commonality associated with each design. By color coding the solutions according to CI, the designers were able to see the clustering of the solutions based on different commonality levels. In Fig. 18.9, the color code from 1 to 7 represents different numbers of commonality variable values needed for the UTC product family. The figure indicates that using three values for the commonality variable, which corresponds to the green color, makes a reasonable balance between Obj1 and Obj2. Using only one commonality variable value across the whole family increases Obj2; see Fig. 18.9—dark blue solutions.

**Fig. 18.10** Decomposition-based product family optimization



There are disadvantages associated with the all-in-one approach. Since the approach lumps all outputs together, it does not consider individual performance explicitly, making it difficult to trade-off performance among individual products. At the same time, typically, it is difficult to scale the problem formulation as the number of product variants increases. Consequently, the decomposition-based approach is an alternative approach to platform family optimization advocated by Khajavirad et al. (2007). This approach considers each individual product performance explicitly, which makes it flexible to accommodate individual product-specific evaluation criteria. Khajavirad et al. (2007) also hypothesized that the decomposition approach is likely to explore global optimality efficiently than the all-in-one approach.

As shown in Fig. 18.10, in the decomposition-based approach, the product family optimization problem is decomposed into two levels: (1) commonality optimization and (2) individual optimization. The commonality optimization determines the optimal platform configuration, while each individual optimization explores design space for each product variant. First, the commonality optimization problem communicates commonality decisions to individual optimizations. Second, an individual optimization problem typically uses the commonality decision as additional constraints, to search for a design that optimizes product performance. Finally, the individual optimization problems communicate corresponding product performance back to the commonality optimization problem.

Typically, product family-based optimization approaches should be able to handle multiple commonality assessments concurrently, for example, potential commonalities for more than one component. According to our experience, these approaches also require less data post-processing.

In the current phase of the research, we have developed optimization formulations for both all-in-one and decomposition approaches. Results shown in Fig. 18.9 were obtained by implementing this formulation on a single processor computer, which were found to significantly inadequate from computational perspective. In the future, we will implement both the formulations on a parallel computing facility, which is needed to solve the UTC product family problem.

**Table 18.2** Comparison of Methods 1, 2, and 3

	Method 1	Method 2	Method 3
Optimal product family solution	Worst	Best	Worst
Ease of use	Best	Moderate	Worst
Interactive capability	Best	Best	Moderate
Suitable for implementation	Best	Moderate	Worst
Training required	Best	Moderate	Worst
Versatility to product family types	Moderate	Moderate	Best
Robustness	Best	Moderate	Moderate

### 18.4.4 Comparison of Methods: Our Experience

This investigation of three methods had a dual purpose: (1) design a UTC product family and (2) determine the extent to which designers would embrace/accept any of the proposed methods. As noted in Sect. 18.2, designers have been hesitant when it comes to commonality selection. Our tenet was that having tools to visualize (1) the product family data, and (2) trade-offs from sharing common components, would help overcome these fears. Although we cannot present specifics of the product family due to its proprietary nature, we share our experiences and observations at a relatively high level. Table 18.2 summarizes our findings, and specific aspects of each method are discussed in the ensuing paragraphs. We note that the comparison of these methods is based on our experience with the UTC product family only. Because of this limited scope, our findings primarily provide insight into the advantages and disadvantages of each method, not firm conclusions.

#### 18.4.4.1 Discussion of Method 1

As shown in Table 18.2, Method 1 fared surprisingly well in every category, yet it did produce poor-quality designs (poor trade-off resolution between different objectives). As such, the designs obtained from Method 1 were substantially inferior to those found using Method 2. Results indicate that the search strategy—exhaustive sampling—used in Method 1 failed to explore the product family design space sufficiently. Increasing the number of samples is an intuitive approach to improve design space exploration. Unfortunately, such an increase is ad hoc and may not always ensure improved design space exploration. Consequently, Method 1 was less attractive compared to Method 2.

#### 18.4.4.2 Discussion of Method 2

According to our experience, Method 2 was found to be the most attractive of all three, which was unexpected. As shown in Table 18.2, Method 2 was found to be the best method (and the only so far) for obtaining designs with superior trade-off resolution (all objectives show simultaneous improvement compared to other methods),

which is the underlying premise of this research. Because of the visualization-aided commonality selection, Method 2 is very interactive, which proved to be its biggest strength. However, Method 2 does require (1) availability of optimization tool and (2) a formal training to the designers in the use of such tools. On the positive side, for our application, Method 2 does not require specialized computational facilities (e.g., parallel computing). Interestingly, once the optimization was complete, designers were able to identify promising commonality options in a relatively short time (few minutes) using the visualization tools.

As we recall, Method 2 uses NSGA-II as an optimization algorithm. Typically, NSGA-II requires customizing some of its parameters, such as population size and number of generations. For the UTC product family design problem, significant convergence infeasibility required a population size of 300 and at least 25 generations to generate a uniform Pareto frontier for all product variants. Since the simulation models were not computationally expensive, we could execute them overnight, providing new data in the morning. More computationally intensive simulations will require other strategies (e.g., approximations, surrogate models) when using individual product optimization as advocated in Method 2.

We demonstrated Method 2 on a three-objective problem (objective 1, objective 2, and commonality). At times, the product family may involve more than three objectives, in which case more visualization windows will have to be investigated simultaneously. Typically, the decision-making complexity increases rapidly with the number of objectives in any optimization problem, and so will in the case of Method 2. However, the visualization-assisted interactive decision-making aspect of Method 2 is expected to lower the complexity of decision making in multi-objective design space. Extending Method 2 to problems with multiple objectives in the future would be important to understand decision-making complexities associated with this method.

Overall, Method 2 was found to be the most suitable because it is (1) simple yet effective for commonality selection, (2) capable of finding product family designs, and (3) suitable for implementing in an environment where designers have limited knowledge of formal optimization techniques.

#### **18.4.4.3 Discussion of Method 3**

Method 3 is the most computationally intensive method of all three based on our experience. The significant increase in computational expense can be attributed to several factors. For instance, the convergence infeasibility of the entire product family is nearly 100 % for the all-in-one approach, as all of the products in the family are handled simultaneously in a single optimization problem. Thus, for this UTC problem, Method 3 needs substantially larger population and generation sizes. To date, population sizes of 500 running for more than 200 generations have failed to yield an optimal solution or a solution with better trade-off resolution than other methods. It is important to note that the computational cost of Method 3 is a magnitude higher than that of Method 2. In the case of the decomposition strategy, it involves solving 11 individual optimization problems in a single iteration of

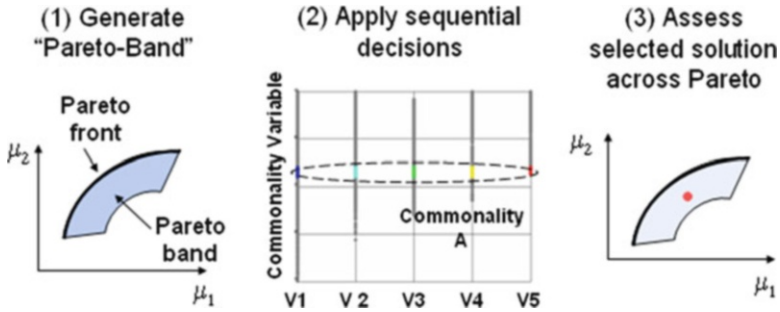


Fig. 18.11 Proposed Pareto band approach

commonality optimization. The computational cost of decomposition appears to be even higher than all-in-one approach. Also, problem formulation and algorithm settings require specialized training, which does not bode well for user adoption.

### 18.4.5 Introduction of the Pareto Band Concept

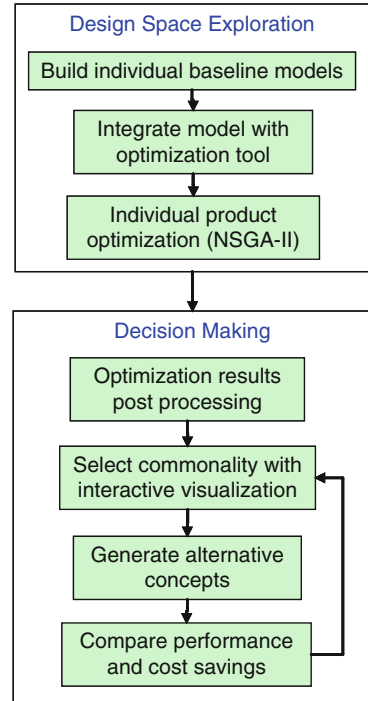
An interesting offshoot of Method 2 has been the identification of a new concept, which we refer to as the Pareto band (see Fig. 18.11). In particular, when reviewing the product families resulting from Method 2, we observed that the product family solutions typically lie in a “band” around individual Pareto frontiers, and typically not on the frontiers. The width of this band indicates the trade-off between commonality and performance—the larger the band, the larger the trade-off as shown in Fig. 18.11. The concept is similar to the *design bandwidth* idea advocated by Claesson and Berglund (2005). However, it works in reverse in that it is driven by the range of solutions that is obtainable in the objective space versus the range of bandwidth one has in the design space.

We are continuing to investigate this finding in more detail to understand its implications better, with the most notable being a potentially new objective for product family optimization, namely, targeting a width of this band to balance the trade-off between commonality in the family and the individual product performance. As such, it is our tenet that the Pareto band approach may find its application in other fields as well, such as robust optimization of a single product.

## 18.5 Implications of Commonality Selection Process

After reviewing all three methods for the UTC product family design problem, the individual product optimization plus visualization method (Method 2) appears to be the most suitable one for adoption with UTC’s business units. The details of the commonality decision-making process are shown in Fig. 18.12 and elaborated as follows.

**Fig. 18.12** Commonality decision-making process



1. Build individual product models using appropriate simulation tools.
2. Integrate each model with an optimization tool.
3. Conduct individual product multiple-objective optimization. In this work NSGA-II is chosen as the optimization algorithm.
4. Post-process results from individual optimization such that the final data file is ready for interactive commonality selection.
5. Apply data filtering and visualization to make commonality selection.
6. Generate alternative concepts based on the desired level of commonality and performance.
7. Compare performance of the new concepts against current baseline designs.

The designers typically need to iterate between Steps 5 and 7 until they are satisfied with the commonality selection and the resulting product family.

By working closely with designers and applying the commonality selection standard work for the UTC product family design, we have identified three new product concepts. The current baseline design uses four commonality variable values, while the new product concepts have either two or three commonality variable values, as shown in Fig. 18.13. In Fig. 18.13,  $C_i$  represents the  $i$ th coil and  $C_i + C_i$  indicates that the  $i$ th coil has been used twice.

The cost savings per year of the new product concepts relative to the current baseline designs are shown in Fig. 18.14. The reported cost savings are only from

Variants	Concept 1	Concept 2	Concept 3	Baseline
	(2 components)	(2 components)	(2 components)	(4 components)
1	C1	C3	C3	C5
2		C4	C4	C4
3				
4				
5				
6	C2	C3 + C3	C3 + C3	C1
7	C1 + C1	C4 + C4	C2 + C2	C2
8	C2 + C2	C4 + C4	C2 + C2	C2
9				
10				
11				

Fig. 18.13 Commonality concepts selected using Method 2

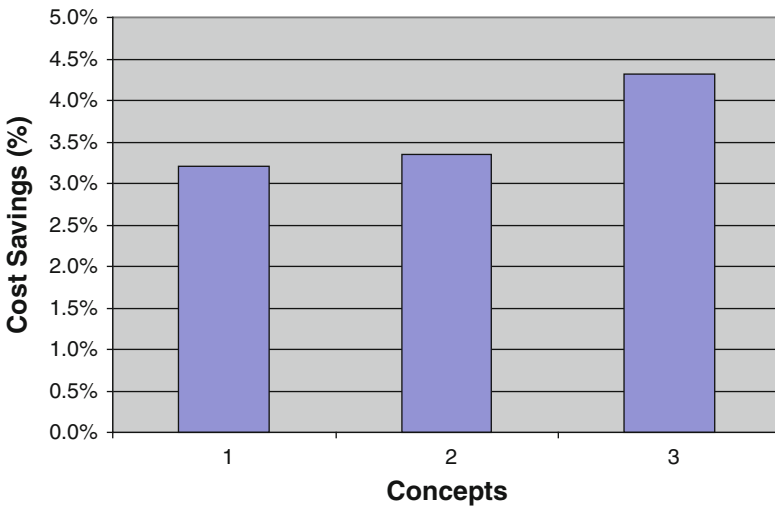


Fig. 18.14 Estimated yearly cost savings for UTC product family

the direct material cost savings. Additional cost savings are expected from reduction of design/development time, qualification tests, supplier volume discount, inventory management, etc.

### 18.6 Closing Remarks

In this chapter, we presented the product family optimization research conducted at the United Technologies Research Center and its application to a UTC product family. Recognizing the challenges associated with product family design with

regard to design space exploration and commonality selection, this work combines optimization-visualization strategies to facilitate product family design. Three different methods are compared and contrasted, and recommendations are given to support UTC's current product family design.

This chapter recognizes gaps occurring in many engineering design practices, i.e., the designer's experience and adoption of design optimization methods. Many times designers rely solely on their own domain knowledge to make design decisions, rather than seeking help from optimization or other advanced design methods. One of the rationales for such gaps is that designers typically do not have visibility inside the result generation process of "black box" optimization methods. Without the capability to view solutions and visualize the trade-offs, they are often hesitant to trust such results.

The individual product optimization and interactive visualization method (Method 2) attempts to bridge this gap by giving designers freedom to interactively make commonality selection and, perhaps more importantly, visualize its effect on individual product performance. As such, the method attempts to visually present the effect of designers' commonality selection on two key entities: (1) the gain from commonality and (2) performance losses incurred in the variants.

To realize the complete benefits of the individual product optimization and visualization method, the following improvements are warranted in the future: (1) the visualization tool, which requires substantial data processing at present, needs to be customized for visualizing the product family design space; (2) at times, the absolute benefits and corresponding performance losses of commonality decisions are not obvious until the designer post-processes the concepts. Development of additional subroutines is required to eliminate the need for post-processing outside the framework of Method 2; and (3) the interactivity only happens after the design spaces for all the products have been explored by optimization techniques. It would be worthwhile to explore the impact of providing the designers access to optimization-assisted design exploration process.

We expect that by gaining such access, designers might be able to guide the design space exploration process towards regions of interest. However, on a cautious note, such an approach will (1) put additional burden on designers and (2) not expose designers to the entire design space prior to decision making, which could potentially constrain the exploration in a narrow space. As such, ATSV currently offers visual steering capabilities to let the user guide the exploration process for a single product (Stump et al. 2009). However, further development work is needed to enable the aforementioned interactivity for the entire product family. Potentially, such interactivity during individual optimization can also be integrated within the decomposition-based approach, which could align with our goal of developing completely interactive product family optimization approaches.

Future work also includes developing robust platform optimization approaches that account for system variability, e.g., uncertain operating conditions, such that the product family designs are less sensitive to these variations. Development of robust platform optimization method would require combining robust design methods with platform optimization approaches.



**Acknowledgments** Dr. Simpson acknowledges support from the National Foundation under Grant No. CMMI-0620948. Any opinions, findings, and conclusions or recommendations presented in this chapter are those of the authors and do not necessarily reflect the views of NSF.

## References

- Balling R (1999) Design by shopping: a new paradigm? In: Proceedings of the third world congress of structural and multidisciplinary optimization (WCSMO-3), University at Buffalo, Buffalo, NY, pp 295–297
- Claesson A, Berglund F (2005) Design bandwidth. In: 2005 innovations in product development conference – product families and platforms: from strategic innovation to implementation, Cambridge, MA
- Collier DA (1981) The measurement and operating benefits of component part commonality. *Decision Sci* 12(1):85–96
- de Weck O (2005) Determining product platform extent. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design: methods and applications. Springer, New York, pp 241–301
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
- Eddy J, Lewis K (2002) Visualization of multi-dimensional design and optimization data using cloud visualization. In: ASME design engineering technical conferences – design automation conference, Montreal, QC, Canada. ASME, Paper No. DETC02/DAC-02006
- Fujita K (2005) Product variety optimization. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design: methods and applications. Springer, New York, pp 186–224
- Halman JIM, Hofer AP, van Vuuren W (2005) Platform-driven development of product families: linking theory with practice. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design: methods and applications. Springer, New York, pp 27–47
- Khajavirad A, Michalek J (2007a) A single-stage gradient-based approach for solving the joint product family platform selection and design problem using decomposition. In: ASME design engineering technical conferences – design automation conference, Las Vegas, NV. ASME, DETC2007/DAC-35611
- Khajavirad A, Michalek J (2007b) An extension of the commonality index for product family optimization. In: ASME design engineering technical conferences – design automation conference, Las Vegas, NV. ASME, DETC2007/DAC-35605
- Khajavirad A, Michalek J, Simpson TW (2007) A decomposed genetic algorithm for solving the joint product family optimization problem. In: 3rd AIAA multidisciplinary design optimization specialist conference, Honolulu, HI. AIAA, AIAA-2007-1876
- Koch PN, Evans JP, Powell D (2002) Interdigitation for effective design space exploration using iSIGHT. *Struct Multidiscip Optim* 23(2):111–126
- Lutz RA (1998) *Guts: the seven laws of business that made Chrysler the world's hottest car company*. Wiley, New York
- Martin MV, Ishii K (1997) Design for variety: development of complexity indices and design charts. *Advances in design automation*, Sacramento, CA. ASME, Paper No. DETC97/DFM-4359
- Messac A, Martinez MP, Simpson TW (2002) A penalty function for product family design using physical programming. *ASME J Mech Des* 124(2):164–172
- Meyer MH, Lehnerd AP (1997) *The power of product platforms: building value and cost leadership*. The Free Press, New York, NY
- Miller S (1999) VW sows confusion with common pattern for models – investors worry profits may suffer as lines compete. *Wall Street J* A.25

- Robertson D, Ulrich K (1998) Planning product platforms. *Sloan Manage Rev* 39(4):19–31
- Simpson TW (2005) Methods for optimizing product platforms and product families: overview and classification. In: Simpson TW, Siddique Z, Jiao J (eds) *Product platform and product family design: methods and applications*. Springer, New York, pp 133–156
- Simpson TW, Martins JRRR (2011) Multidisciplinary design optimization for complex engineered systems: report from a national science foundation workshop. *ASME J Mech Des* 133(10):101002, 10 pages
- Simpson TW, Brennan S, Slingerland LA, Bobuk A, Logan D, Reichard K (2012) From user requirements to commonality specifications: an integrated approach to product family design. *Res Eng Des* 23(2):141–153
- Slingerland LA, Bobuk A, Simpson TW (2010) Product family optimization using a multidimensional data visualization approach. In: 13th AIAA/ISSMO multidisciplinary analysis and optimization conference, Fort Worth, TX. AIAA, AIAA-2010-9031
- Stump G, Lego S, Yukish M, Simpson TW, Donndelinger JA (2009) Visual steering commands for trade space exploration: user-guided sampling with example. *ASME J Comput Inf Sci Eng* 9(4):044501, 10 pages
- Thevenot HJ, Simpson TW (2006) Commonality indices for product family design: a detailed comparison. *J Eng Des* 17(2):99–119
- Winer EH, Bloebaum CL (2001) Visual design steering for optimization solution improvement. *Struct Optim* 22(3):219–229

# Chapter 19

## Developing and Assessing Commonality Metrics for Product Families

Michael D. Johnson and Randolph E. Kirchain

**Abstract** To be competitive in today's global economy, firms must deliver more products that are viable in the marketplace for shorter times. The use of product families allows firms to meet these needs in a cost-competitive manner. The determination of which components to share and which should be unique is very important to the development of product families. Commonality metrics are presented with the goal of assessing (at the early stages of development) the ability of the product family to reduce costs. The methodology of process-based cost modeling is applied to project product development, fabrication, and assembly costs in both the standalone and shared situations. A case study of two automotive instrument panel beams is analyzed. Linear regression analysis shows that when compared to total cost savings, a simple piece commonality metric and a fabrication investment-weighted metric have higher  $R^2$ 's than a mass or piece cost-weighted metric. When correlated to fixed cost savings, the fabrication investment-weighted metric has the highest  $R^2$  (0.62) and is significant at the 0.025 level. Fixed cost savings are proposed as the desired quantity for assessing product family efficiency.

---

An earlier version of this chapter appeared in M. D. Johnson and R. Kirchain (2010) Developing and Assessing Commonality Metrics for Product Families: A Process-Based Cost-Modeling Approach, IEEE Transactions on Engineering Management, 57(4): 634–648 (© IEEE 2010), reprinted with permission.

M.D. Johnson (✉)  
Department of Engineering Technology and Industrial Distribution,  
Texas A&M University, College Station, TX, USA  
e-mail: [johnson@entc.tamu.edu](mailto:johnson@entc.tamu.edu)

R.E. Kirchain  
Engineering Systems Division, Massachusetts Institute of Technology,  
Cambridge, MA, USA

## 19.1 Introduction

The goal of most firms is to deliver products that satisfy customer needs. In recent history, most of these products have been mass-produced over a number of years. This allowed for economies of scale in manufacturing and for development costs and manufacturing investments to be spread over many units of production. For many firms, market conditions have changed and made it challenging to hold to these practices. Specifically, several authors have documented a trend toward increased product variety and shorter product lifetimes (Pine 1991, 1993; Schonberger 1987; Uzumeri and Sanderson 1995; von Braun 1990). These two practices put pressure on firms to develop an increasing array of products that are only viable in the market for a short period. Given limited resources, the importance of development and production efficiency is increased. Although many strategies exist to address this challenge, one that has been shown to be powerful for some types of goods is the use of product platforms and product families (Gonzalez-Zugasti et al. 2000; Muffatto 1999; Meyer and Lehnerd 1997; Ulrich 1995; Gupta and Souder 1998).

A product platform is a set of subsystems from which derivative or variant products can be developed and manufactured (Meyer and Lehnerd 1997). These variant products combine both shared and unique components to perform a given function. The product family is the group of variants that are derived from a given platform (Gonzalez-Zugasti et al. 2000). Because of the scope of their impact, platform and product family decisions occur early in the development process and are inherently encompassing. Platforming decisions are regarded as among the most important in the development process (Robertson and Ulrich 1998). As such, it is critical to have an analytical measure of the effectiveness of a given product family design decision (Jiao and Tseng 2000; Nobelius and Sundgren 2002). To be useful, such a metric must be diagnostic (i.e., correlated to some desired quantity) and analytically feasible against the limited information indicative of early design (even if this information represents rough estimates). The vast majority of research into platform-effectiveness metrics has focused on metrics of commonality (the amount of component sharing among variants) and their correlation to cost (Fixson 2007). This work presents a set of commonality metrics and assesses their correlation to a particular element of their effectiveness, the cost savings resultant from component sharing, specifically savings in development and manufacturing costs. A small-n case study is used to explore the correlation between these metrics and cost savings and to develop hypotheses regarding the characteristics of an ideal metric; case studies provide benefits when developing such hypotheses (Eisenhardt 1989). The metrics and models presented use data typically available at the early stages of the development process. This allows for a feasible and tractable methodology.

## 19.2 Commonality and Platform Literature

Several researchers have investigated the benefits and drawbacks of product platforming and the use of product families. These benefits include reduced production and development costs (Muffatto 1999; Meyer and Lehnerd 1997; Ulrich 1995; Gupta and Souder 1998; Park and Simpson 2008) and shortened development time (Gonzalez-Zugasti et al. 2000; Krishnan and Gupta 2001; Muffatto 1999). The reuse of subassemblies and components can reduce technological risk (Clark and Fujimoto 1989; Rosenthal and Tatikonda 1992) and increase labor productivity (MacDuffie et al. 1996). Ultimately, proper design of a product family can have significant technological and economic advantages (Maier and Fadel 2001).

In addition to development and manufacturing cost-related benefits, several authors have noted the operational benefits of component commonality or conversely of the reduction in component variety. Some of these operational benefits are inventory related, such as decreased buffer inventory level (Tsubone et al. 1994), reduced holding costs (Vakharia et al. 1996), and risk pooling benefits (Thonemann and Brandeau 2000; Hillier 2002). In some cases the order pooling benefits of common components dominate their risk pooling benefits (Hillier 2002). Other operational benefits include increased quality (McDermott and Stock 1994), improved operational flexibility (Maskell 1991), and reduced spare parts provisioning costs (Kranenburg and Van Houtum 2007). Clearly, these impacts on operations can provide important economic benefits as well. Although the analysis here is limited to changes in development and direct manufacturing cost, the implications of these other cost effects will be discussed qualitatively.

There can also be drawbacks to the use of a platform strategy. Common components can lead to extensive change requirements when a widely used component is altered (Ho and Li 1997) and cause coordination difficulties (Clark and Fujimoto 1989). Inherently, common components reduce product variety (albeit sometimes in ways not consumer perceivable). Nevertheless, the requirement of reduced variety constrains the design space (Clark and Fujimoto 1989). In some cases, this constraint leads to excess functionality (Thomas 1992), reduced performance (Nelson et al. 2001) or client responsiveness (Oshri and Newell 2005) and even lower perceived quality (Krishnan and Gupta 2001; Gonzalez-Zugasti et al. 2000; Yu et al. 1999; Kim and Chhajed 2001). These cost and revenue implications of reduced variety can be serious.

Given the putative benefits of component commonality, it has been proposed and used as a quantitative design goal (Nobelius and Sundgren 2002; Skold and Karlsson 2007). To effectively guide design, it has been proposed that commonality metrics should be simple and relevant (Maskell 1991). Nevertheless, despite widespread recognition of the importance of mapping the relationship between commonality and cost (Fixson 2005; Thonemann and Brandeau 2000; Simpson et al. 2006), no widely accepted relationship has been established. Clearly, the use of commonality metrics for assessment increases the importance of characterizing such a relationship.

To establish an effective, quantitative measure of component commonality, several researchers have proposed alternative commonality metrics and indices to be used to guide product family design decisions. Moscato (1976) proposes a relative commonality metric based on the ratio of actual component (or part) commonality to maximum commonality. Collier (1981) proposes a commonality index that relates the number of parent items to the number of distinct component parts and shows total cost decreases with higher degrees of commonality. Guerrero (1985) uses Collier's commonality index and shows the benefits of commonality given unknown demand. Wacker and Treleven (1986) propose a bounded measure (0–1) of part commonality as well as suggesting different types of part commonality that can be measured. Additional commonality metrics based on ratios of system component variety are also proposed (Thomas 1992; Tsubone et al. 1994). Jiao and Tseng (2000) add production volume and the cost of components to Collier's commonality index. Kota et al. (2000) develop a product line commonality index and show that the Sony Walkman line of products performs better (according to this index) than competitors. The Sony Walkman is an oft-cited case of superior product family management (Sanderson and Uzumeri 1995). Martin and Ishii (1996, 1997) use a commonality metric based on the ratio of unique components to total components in the product family as part of their design for variety methodology. Blecker and Abdelkafi (2007) propose a commonality metric for use in mass customization environments. Thevenot and Simpson (2006) present a comprehensive commonality metric which takes into account manufacturing process, material, assembly scheme, and initial cost. Fixson (2007) provides an extensive review of research in the area of commonality; as mentioned previously, the vast majority of research into commonality relates to its effects on cost.

Given the key role of commonality decisions in the development of product families (Alizon et al. 2007; Zacharias and Yassine 2008), it is important that the proper metric be used when assessing alternative product family concepts. While a universal relationship between commonality and cost has yet to be established (Labro 2004), the goal of this work is to establish what information within a commonality metric is most effective at mapping the link between component commonality and resultant development and manufacturing cost savings. This is accomplished by evaluating a set of metrics that meet the above requirements; the evaluation criterion is the correlation of the metric to cost savings. These metrics include a simple piece-count-based metric, component characteristic-weighted metrics, and a version that accounts for relative production volume. Process-based cost models of component development, fabrication, and assembly are used to project costs in both the product family and single-variant cases. These projections are used to explore the correlation between the metrics and the cost savings resulting from component sharing. The cost modeling methodology and commonality metrics are described in the next section.

## 19.3 Modeling and Assessment Methods

This section presents the set metrics which were assessed and the cost modeling methodology used to project the consequences of platformed and standalone design alternatives. The metric assessment method is also detailed in this section.

### 19.3.1 Proposed Commonality Metrics

The general commonality metric proposed here ( $C$ ) is defined as the ratio between the number of components shared and the number of components that could be shared in the given product family; in its basic form it is piece-based commonality ( $C_{Piece}$ ). It is similar to the metric presented by Moscato (1976) in that it used a ratio of actual commonality to maximum commonality. This ratio is calculated for each line item in the combined bills of materials for all variant products in the family being analyzed. These quantities are summed and then divided by the total number of line items. This metric (and all metrics presented in this section) has a minimum value of 0 (no commonality) and a maximum value of 1 (full commonality). This metric is similar to the bounded version of Collier's (1981) commonality metric presented by Wacker and Treleven (1986). Thevenot and Simpson (2006) propose a metric with a similar commonality criterion that can also be used with a simple nonhierarchical bill of materials; however, their metric attempts to be more comprehensive (by convoluting numerous component characteristics). In contrast, the metrics explored here are more focused, allowing for a more targeted assessment of the relationship between the metric and cost savings.

Before introducing any measure of product family commonality, it is necessary to establish a formal definition of a part's shared status, which will be represented subsequently as  $\gamma_i$ . This is required because in real-world products, shared subassemblies and components do not have to be identical. Some components may share the majority of forming production steps, differing only because of limited finishing operations such as trimming or drilling of holes. For the purpose of this work, a component or assembly was considered shared if it used the same primary forming tooling and equipment as another part in the family.

It is worth discussing this definition both in terms of its ability to be assessed at early design stages and relative to other definitions presented in the literature. Regarding the former, in the authors' experience, part designers quickly learn those design modifications that require tooling changes. As such, most would be able to reasonably assess whether two variants will require additional tooling. Of the other definitions for commonality presented in the literature, one comprehensive scheme can be found in the work of Kota et al. (2000), who propose three criteria by which to judge whether a component is common or not—(1) size and shape, (2) material and process, and (3) assembly and fastening scheme. For most cases, parts that differ in any of Kota's three criteria will require new forming tools and, therefore, would be considered distinct in this work. However, other than for criteria two (i.e., differing production process), this rule will not always pertain.

**Table 19.1** Example bill of materials for a product family

Parts ( <i>i</i> ) <i>d</i> = 4	Variants ( <i>j</i> ) <i>m</i> = 3			Key commonality metric intermediate calculations				
	V1	V2	V3	$\phi_i$	Parts shared ( $\gamma_{ij}$ )	$\frac{\sum_{j=1}^m \gamma_{ij}-1}{m-1}$	$\phi_i \frac{\sum_{j=1}^m \gamma_{ij}-1}{m-1}$	$\frac{\sum_{j=1}^m PV_{ij}-PV_{\min}}{PV_{\text{Tot}}-PV_{\min}}$
A	X	X	X	5	2	1.0	5.0	1.00
B		X		3	0	0.0	0.0	0.13
C	X	X		2	1	0.5	1.0	0.80
D		X	X	7	1	0.5	3.5	0.33
Production volume	100	50	30					

For example, components made by a number of processes, including stamping, extrusion, or roll forming, that differ only in one dimension can sometimes be made using a common forming tool in combination with different trimming requirements. In considering these differences between Kota’s definition and the one employed here, it is important to note that the purpose of this work is to assess the economic impact of alternative product family strategies. Since secondary processing costs such as drilling and finishing are usually much less than major forming costs, components that share major forming tooling are considered common. In other cases where complexity or holding costs are being assessed, part differences such as finishes and hole patterns might become more relevant.

Given these definitions of commonality, the simple piece-based commonality metric evaluated in the work is calculated as:

$$C_{\text{Piece}} = \frac{\sum_{i=1}^d \left( \frac{\sum_{j=1}^m \gamma_{ij}-1}{m-1} \right)}{d} \tag{19.1}$$

where  $\gamma$  is a binary variable:  $\gamma = 1$ , if variant  $j$  contains component  $i$ , and  $\gamma = 0$ , if it does not.

The total number of product variants is  $m$  and  $d$  is the number of distinct items in the bill of materials. One is subtracted from the sum of variants containing a component and the total number of variants to account for the first variant that contains this component; this allows for the resulting ratio to be the number of variants sharing a component to the number of variants that could share that component. This methodology can be extended to calculate the commonality of subassemblies in a product family by replacing the component line items with subassembly line items. An example bill of materials for a product family as well as some key sharing calculations for this product family is shown in Table 19.1. In this example,  $m = 3$ ,  $d = 4$ , and  $C_{\text{Piece}}$  is equal to 0.5 (see Table 19.2).



**Table 19.2** Calculation of four commonality metrics for the product family presented in Table 19.1

$C_{Piece}$	$C_{\phi}$	$C_{PV}$	$C_{PV/\phi}$
$\sum_{i=1}^d \left( \frac{\sum_{j=1}^m \gamma_{ij}^{-1}}{m-1} \right)$	$\sum_{i=1}^d \left( \frac{\sum_{j=1}^m \gamma_{ij}^{-1}}{\phi_i \cdot m-1} \right)$	$\sum_{i=1}^d \left( \left( \frac{\sum_{j=1}^m \gamma_{ij}^{-1}}{m-1} \right) \left( \frac{\sum_{j=1}^m PV_{ij} - PV_{\min}}{PV_{Tot} - PV_{\min}} \right) \right)$	$\sum_{i=1}^d \left( \phi_i \left( \frac{\sum_{j=1}^m \gamma_{ij}^{-1}}{m-1} \right) \left( \frac{\sum_{j=1}^m PV_{ij} - PV_{\min}}{PV_{Tot} - PV_{\min}} \right) \right)$
$d$	$\sum_{i=1}^d \phi_i$	$d$	$\sum_{i=1}^d \phi_i$
0.5	0.56	0.39	0.41

Research shows that relevant commonality metrics reflect the relative production volumes as well as some measure of cost and/or complexity of the components being analyzed (Jiao and Tseng 2000; Wacker and Treleven 1986). Thevenot and Simpson (2006) include piece cost variable in their commonality metric. Jiao and Tseng (2000) also include a cost variable in their commonality metric. Their metric requires an analysis of the hierarchical product structure to determine the number of component parents. The following commonality metric includes a flexible weighting parameter  $\phi_i$  and can be used with a single-level combined bill of materials:

$$C_{\phi} = \frac{\sum_{i=1}^d \left( \phi_i \frac{\sum_{j=1}^m \gamma_{ij}^{-1}}{m-1} \right)}{\sum_{i=1}^d \phi_i} \tag{19.2}$$

where  $\phi_i$  can capture the relative importance of component  $i$ , such as its mass, piece cost, or fabrication investment. This measure will be referred to as the  $\phi$ -weighted commonality metric ( $C_{\phi}$ ), with  $\phi$  specified (e.g.,  $C_{\phi=Mass}$  for the mass-weighted commonality metric). When multiple components are needed per variant, the weighting factor is scaled by the number needed per variant. For the example product represented by Table 19.1,  $C_{\phi} = 0.56$  (see Table 19.2).

While Jiao and Tseng (2000) include the volume of products in their commonality metric, the metric proposed below takes production volume and relative component importance into account while maintaining simple 0 (no commonality) and 1 (full commonality) end points:

$$C_{PV/\phi} = \frac{\sum_{i=1}^d \left( \phi_i \left( \frac{\sum_{j=1}^m \gamma_{ij}^{-1}}{m-1} \right) \left( \frac{\sum_{j=1}^m PV_{ij} - PV_{\min}}{PV_{Tot} - PV_{\min}} \right) \right)}{\sum_{i=1}^d \phi_i} \tag{19.3}$$

**Table 19.3** Descriptions of proposed commonality metrics

Metric	Description
$C_{Piece}$	The piece-based metric is calculated using whether a part is shared
$C_{\phi=Mass}$	The mass-weighted metric is calculated based on the relative mass of a component in relation to the sum of the mass for the entire product family
$C_{\phi=Cost}$	The cost-weighted metric is based on the fabrication cost per piece (assuming component sharing)
$C_{\phi=Invest}$	The investment-weighted metric is based on the fabrication investment required for a component (assuming component sharing)
$C_{PV}$	The production volume-weighted metric is calculated using the relative production volumes required for each variant
$C_{PV/\phi=Invest}$	The production volume-/investment-weighted metric combines relative production volume and fabrication investment weightings

where  $PV_{ij}$  is the production volume for variant  $j$  for component  $i$  (a unique line in the bill of materials).  $PV_{Tot}$  is the sum of the production volumes for all variants in the product family.  $PV_{min}$  is the production volume for the product family variant that has the minimum production volume. The quantity derived using Eq. (19.3) will be referred to subsequently as production volume/ $\phi$ -weighted commonality measures ( $C_{PV/\phi}$ ). For the case where all parts are weighted equally:

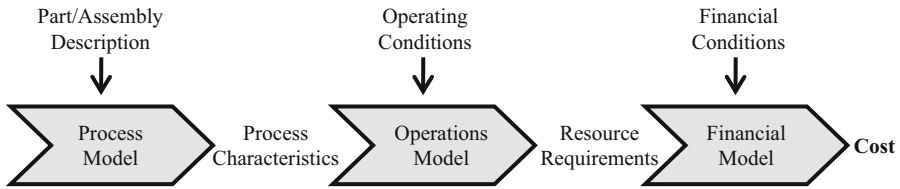
$$\phi_i = \frac{1}{\sum_{i=1}^d \phi_i} \forall i \quad (19.4)$$

Equation (19.3) collapses to provide an analogous production volume adjusted version of the piece-based commonality metric that is referred to subsequently as the production volume-weighted commonality metric ( $C_{PV}$ ). For the example product represented by Table 19.1,  $C_{PV/\phi} = 0.41$  and  $C_{PV} = 0.39$  (see Table 19.2).

Commonality metrics will be assessed based on their correlation with cost savings (as measured by  $R^2$ ). The proposed metrics are summarized in Table 19.3 along with a brief description of each. The next section details the method used to project costs and costs savings.

### 19.3.2 Cost Modeling Methodology

The purpose of the cost models used in this work is to project development and manufacturing costs and capture the effects of component commonality on these costs. These projections are used to assess the commonality metrics proposed above. Much research has been conducted with the goal of trying to determine the cost of products; this has been done for many purposes (e.g., Ulrich and Pearson 1998; Zhang and Tseng 2007; Field et al. 2007; Tu et al. 2007). Specifically, the relationships between product and process characteristics and cost have often been analyzed. Activity-based costing (ABC) is a widely cited method that traces costs to



**Fig. 19.1** Schematic of process-based cost modeling

causal activities and processes (Angelis and Lee 1996; Cleland 2001; Cooper and Kaplan 1988). While ABC methods have been proposed and used for predictive purposes (e.g., Qian and Ben-Arieh 2008), they require adherence to strict conditions (Noreen 1991). When these are not met, ABC methods can produce significant error (Noreen and Soderstrom 1994). Given the requirement of strict proportionality between activities and cost pools (requiring only linear cost functions with zero intercepts) and the inability to capture dependencies between products in ABC costing methods, which is fundamental to determining the cost effects of sharing components (Noreen 1991), process-based cost modeling is proposed for the projection of the cost and cost savings in this work.

Process-based cost modeling (PBCM) is an early stage, generative cost estimation tool that uses various part and process characteristics to project manufacturing, assembly, and product development costs. Process-based cost models for several manufacturing processes exist and have been used to answer numerous research questions around the comparison and selection of materials, processes, and architectures (Field et al. 2007). Process-based cost models are constructed by working backward from cost—the model’s objective—to physical parameters that can be controlled: the model’s inputs. The modeling of cost involves (1) correlating the effects of relevant physical parameters on the cost-determinant attributes of a process (e.g., cycle time, equipment performance requirements), (2) relating these processing attributes to resource requirements (e.g., kg of material, person-hours, number of machines and/or tools), and (3) translating these requirements to a specific cost (Kirchain and Field 2001). The relationship between physical parameters and process characteristics is determined by using physical relationships and/or through statistical analysis.

The inputs required for a PBCM can be broken into four main categories: part and material related, process related, operational, and financial. A schematic of process-based cost modeling can be seen in Fig. 19.1, which shows the three key modeling steps leading from case description through process characteristics to operation characteristics and finally to cost. The specifics of the models vary depending on whether development or manufacturing costs are being projected, but the framework is consistent. In each case the characteristics of a part or assembly are correlated to processing characteristics (e.g., required design time or fabrication and assembly cycle time). These processing requirements are used along with operating conditions, such as the number of work hours and days, to determine resource requirements (e.g., the number of person-hours for design,

the number of tools or machines for manufacturing). Finally, financial conditions (the costs of resources and interest rates) and resource requirements are used to project total cost. These costs are allocated to individual cost categories.

The output for the development cost model consists of the development costs for the stages under consideration. In this case these included detailed design, a formability (or manufacturability) analysis, and the engineering required for the fabrication and assembly processes. For the manufacturing cost models (fabrication and assembly), output cost categories included both variable (labor, materials, and energy) and fixed (tooling, equipment, overhead and maintenance, and building) costs. A full description of process-based cost models for development, fabrication, and assembly is outside the scope of this work; interested readers are directed to Fuchs et al. (2008) and Johnson and Kirchain (2009a) for a general description and to Johnson and Kirchain (2009b) for discussion of the standalone and shared costs presented in this work.

To evaluate the fidelity of the development model results, the model was applied against a number of historic subassembly development projects within the same major automaker from whom data was collected but which were distinct from the projects on which data was collected. The historic projects had each been evaluated and assessed an engineering effort (in person-hours) through conventional accounting approaches within the firm. These independent assessments were compared against model results. All modeled results were within 20 % and most were within 10 % of the firm assessment figures. The results of the part production and assembly models have been evaluated by comparing the resource requirements projected by the models against the resources consumed at actual facilities. For the processes examined in this document, this validation has been carried out across case studies as documented in Roth and Shaw (2002), Cirincione (2008), and Kar (2007).

Because product family decisions inherently involve multiple products, explicit rules are required for allocating common component costs. To address this need, the following nomenclature and analytical structures were developed to analyze costs here. Let the product family  $w$  comprise the set of variants  $V^w$  of which variant  $j$  is a member. Each  $j$  is assumed to be a member of one and only one family,  $w$ . Let  $A^j$  be the set of all components required to produce  $j$ .  $A^j$  is the union of  $\Xi^j$ , the set of components exclusively used by  $j$ , and  $Z^{j \in w}$ , the set of components used by  $j$  as well as at least one other variant in  $V^w$ : ( $A^j = \Xi^j \cup Z^{j \in w}$ ). Finally, let each component  $i$  within the  $w$  be associated with a total production cost,  $C_{i,Total}$ , comprising the cost of parts production, assembly, and/or development.

Firstly, for the product family, the costs of all variants can be determined as if each variant were produced independently, without the benefits of sharing; this is defined as the variant's Standalone Cost,  $X_j$ . These costs can then be summed to determine the cost of the entire product family assuming no sharing; this results in the Standalone Cost of the product family. Using the above notation, it is possible to define the simple Standalone Cost of variant  $j$  as:

$$\text{Standalone Cost}_j = X_j = \sum_{i \in A^j} C_{i,Total} \quad (19.5)$$

The cost of a variant when including the effects of component sharing is calculated as the production volume-weighted ratio of that variant's cost contribution to the total cost for that part or subassembly. The shared cost ( $\Delta_j$ ) of variant  $j$  in platform  $w$  is:

$$\begin{aligned} \text{Shared Cost}_j = \Delta_j &= \text{Exclusive Cost}_j + \text{Total Shared Cost} \times \frac{PV^j}{\sum_{\{l|Z^{w,l} \subseteq A^j\}} PV^l} \\ \Delta_j &= \sum_{i \in \Xi^j} C_{i,Total} + \sum_{i \in Z^j \in w} C_{i,Total} \times \frac{PV^j}{\sum_{\{l|Z^{w,l} \subseteq A^j\}} PV^l} \end{aligned} \quad (19.6)$$

Here, *Exclusive Cost* is the sum of the costs of unique components, those that are not shared with other variants in  $V^w$ , and  $PV$  represents the production volume. In aggregate, the shared cost of a variant is the sum of the costs of unique components and shared costs for common components. These definitions and calculations can be used for individual cost categories (i.e., development, fabrication, or assembly) or for the aggregate cost of all three categories.

As mentioned previously, the economic value of a product family strategy arises from the assumed differential between its standalone costs and the shared costs. To normalize this differential and give a consistent output against which commonality metrics can be compared, the cost savings metric ( $S$ ) was calculated as follows:

$$S^w = \frac{\sum_{k \in V^w} X_k - \sum_{j \in V^w} \Delta_j}{\sum_{j \in V^w} X_j} \quad (19.7)$$

where  $X_j$  is the cost of variant  $j$  assuming that there is no sharing, the standalone cost;  $\Delta_j$  is the shared cost. The cost savings metric ( $S$ ) provides a relevant and objective goal against which commonality metrics can be assessed.

### 19.3.3 Metric Assessment Methodology

As mentioned previously the goal of this work is to map component commonality to resultant cost savings. To do so, simple linear regression was used to assess the relationship between the various commonality metrics and the resultant cost savings. The fit of the linear relationship between a given metric and the projected cost savings ( $S$ ) resultant from component sharing served as the basis of metric performance (with better fit or higher  $R^2$  signaling better metrics). In the following section, a case study is used to explore these relationships.

## 19.4 Case Studies

### 19.4.1 Instrument Panel Beam Comparison

To assess alternative commonality metrics based on their relationship to cost savings, two alternative instrument panel (IP) beam product families were analyzed using the process-based cost modeling methodology (described in the preceding section): (1) a tube-based steel design (common in vehicles today) and (2) a die-cast magnesium design which affords significant parts consolidation.

The designs of both alternatives were developed with the input of designers at a major US automotive OEM. Both are considered functionally equivalent, except for the weight savings afforded by the magnesium design. Although representative of designs used in a mid-sized sedan, these designs do not reflect components within any specific vehicle. The steel IP beam (subsequently denoted steel IP) consisted of a tubular structure with over two dozen brackets attached. The magnesium design comprised a primary die-cast magnesium structure (denoted Mg IP) with two additional unique bracket pairs. Both cases also were designed so that variants one and two shared a larger portion (or the major component in the case of the magnesium design) of parts between themselves than either variant one or variant two shared with variant three. For the two cases, the three variants were assumed to be produced at various production volumes. Table 19.4 provides general operational and financial assumptions made for the purposes of modeling manufacturing and cost. All such inputs are representative of conditions experienced by automotive manufacturers in developed countries, but do not reflect the operating conditions of any specific firm. Table 19.5 details key processing information about the two designs. Processing information for these parts was estimated using the process-based models. Notably, the major die-cast part is projected to have a production rate approximately 2–3 times slower than that of the analogous steel components. Due to the highly proprietary nature of some data used in this work, all cost data is disguised through normalization to protect the industrial sponsor. A summary of the variants used in the case is shown in Table 19.6. All trends shown are consistent and proportional to those found using un-normalized data. Component details and commonality are shown in Tables 19.7 and 19.8. Component sharing specifics were based on input from component designers for both cases. The fabrication investment and per piece cost were projected using process-based cost models described above. The investment data in Tables 19.7 and 19.8 are normalized to a value of 1. Piece cost and investment data are shown assuming component sharing among the three variants.

The projected per part development, fabrication, and assembly costs for variant one of the steel and magnesium designs are presented to provide the reader with the cost structure for the two cases. Figures 19.2, 19.3, and 19.4 show these costs for a standalone variant one. The annual production volume is assumed to be 75,000 units per year. Figure 19.2 shows the development cost breakdown for the two designs. In both cases, development costs are dominated by assembly engineering. The development costs for the steel design are more than six times those of the

**Table 19.4** Operational and financial assumptions

<i>Model inputs</i>	
Annual production volume	75,000 parts/year
Days per year	235 days/year
Wage (including benefits)	\$50/h
Unit energy cost	0.05 \$/kW h
Periodic discount rate	10 %
Indirect workers/direct worker (part fabrication)	0.25
Indirect workers/line (part fabrication)	1
Building unit cost	1,200 \$/m <sup>2</sup>
Product life (tooling life)	5 years
Equipment life	15 years
Building life	40 years
Equipment	Nondedicated
Buildings	Nondedicated
<i>Downtimes</i>	
Hours per day	7 h/day
Worker unpaid breaks	1 h/day
Worker paid breaks	1.2 h/day
<i>Material prices</i>	
Magnesium price	\$3.10/kg
Magnesium scrap price	\$2.30/kg
Steel sheet price	\$0.81/kg
Steel tube price	\$1.30/kg
Steel scrap price	\$0.10/kg

**Table 19.5** Instrument panel beam processing information

Name	Manufacturing process	Reject rate	Trim loss	Melt loss	Cycle time (s)
<i>Steel beam parts</i>					
Upper structure A	Tube bending	0.2 %	5 %	0 %	67
Upper structure B	Tube bending	0.2 %	5 %	0 %	73
Lower structure 1	Tube bending	0.2 %	5 %	0 %	46
Lower structure 2	Purchased tube	N/A	N/A	N/A	N/A
Representative bracket (34 total)	Stamping	1.0 %	20 %	N/A	2
<i>Magnesium beam parts</i>					
Beam structure A	Die casting	1.0 %	2 %	3 %	142
Beam structure B	Die casting	1.0 %	2 %	3 %	153
Representative bracket (4 total)	Stamping	1.0 %	20 %	N/A	2

magnesium design; this can be attributed to the significantly larger number of components. The fabrication cost allocation, shown in Fig. 19.3, for the two designs diverges. The magnesium design is dominated by material cost (due to the high price and large portion of magnesium), while in the steel design costs are

**Table 19.6** Summary of product variants

	Production volume	Unique parts	Standalone cost	Shared cost
<i>Steel beam variants</i>				
Variant 1	75,000	1	\$39.74	\$30.31
Variant 2	25,000	1	\$82.38	\$37.90
Variant 3	50,000	8	\$50.85	\$33.02
<i>Magnesium beam variants</i>				
Variant 1	75,000	0	\$43.14	\$37.80
Variant 2	25,000	2	\$58.79	\$40.46
Variant 3	50,000	1	\$41.03	\$35.27

apportioned more equally among the tooling, material, and other fixed cost categories. The overall fabrication costs for the magnesium design are almost twice those of the steel design. The assembly cost allocation is shown in Fig. 19.4. The assembly costs in both cases are largely comprised of the equipment (under Other Fixed) required to assemble the IP beams. Tooling and labor also account for a large portion in both cases. The assembly cost for the steel design is almost twice that of the magnesium design. Overall, the total projected per part cost for the steel design is less than that of the magnesium design under the modeled conditions.

### 19.4.2 Case Results

The cost models in this work are used to project costs and cost savings resulting from shared components. The costs and commonality metrics for the analyzed product families are calculated using the methodology presented above. In addition to the product families containing all three variants, product families of variant pairs are also analyzed. The data for which components are shared among product variants are shown in Tables 19.7 and 19.8. Figure 19.5 shows the projected costs for the steel IP design assuming no component sharing (standalone) and assuming component sharing (shared). The results show significant cost savings in the development and assembly cost categories for variants one and three; for variant two cost savings are distributed among all three cost categories (due to this variant's low standalone production volume). Figure 19.6 shows the projected standalone and shared costs for the magnesium product family containing all three variants. In all three variants, the majority of cost savings arise from reduced development and assembly costs as a result of shared components and assembly processes. This is due to the sharing strategy and the significant portion of the fabrication cost due to material cost.

The proposed commonality metrics were calculated for both three variant product families as well as for the two variant product families. The results of these calculations and the projected total and fixed cost savings for each product



**Table 19.7** Component and commonality information for magnesium IP beam

Part name	Total production			Part mass (kg)	Piece cost (\$)	Fabrication investment	Piece sharing	Mass sharing	Piece cost sharing	Investment sharing	Production volume sharing	Investment/production volume sharing
	Variant 1 (75,000)	Variant 2 (25,000)	Variant 3 (50,000)									
Beam structure—A	X	X	100	8.1	\$32.47	0.617	0.5	4.05	\$16.24	0.308	0.3	0.185
Beam structure—B			50	7.3	\$29.53	0.289	0	0	\$0.00	0.000	0	0.000
Bracket 1—A (x2)	X	X	250	0.20	\$0.35	0.035	0.5	0.2	\$0.35	0.018	0.4	0.014
Bracket 1—B (x2)		X	50	0.22	\$0.66	0.012	0	0	\$0.00	0.000	0	0.000
Bracket 2—A (x2)	X	X	250	0.15	\$0.31	0.034	0.5	0.15	\$0.31	0.017	0.4	0.014
Bracket 2—B (x2)		X	50	0.17	\$0.60	0.012	0	0	\$0.00	0.000	0	0.000

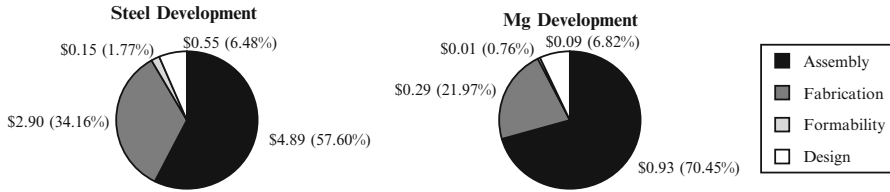
**Table 19.8** Component and commonality information for steel IP beam

Part name	Total production volume ('000s)				Part mass (kg)	Piece cost (\$)	Fabrication investment	Piece sharing	Mass sharing	Piece cost sharing	Investment sharing	Production volume sharing	Investment/production volume sharing
	Variant 1 (75,000)	Variant 2 (25,000)	Variant 3 (50,000)	Variant 4 (50,000)									
Air bag bracket 1	X	X	X		0.34	\$0.52	0.032	1	0.34	\$0.52	0.032	1	0.032
Air bag bracket 2	X	X	X		0.32	\$0.51	0.032	1	0.32	\$0.51	0.032	1	0.032
Air bag bracket 3—A	X	X			0.26	\$0.53	0.024	0.5	0.13	\$0.26	0.012	0.3	0.007
Air bag bracket 3—B			X		0.26	\$0.72	0.017	0	0.00	\$0.00	0.000	0	0.000
Compartment hanger bracket	X	X	X		0.25	\$0.44	0.031	1	0.25	\$0.44	0.031	1	0.031
Fuse bracket 1	X	X	X		0.30	\$0.49	0.032	1	0.30	\$0.49	0.032	1	0.032
Fuse bracket 2	X	X	X		0.81	\$0.78	0.035	1	0.81	\$0.78	0.035	1	0.035
Inner drivers side bracket	X	X	X		0.30	\$0.49	0.032	1	0.30	\$0.49	0.032	1	0.032
Inner passenger bracket—A	X	X			0.30	\$0.56	0.025	0.5	0.15	\$0.28	0.012	0.3	0.007
Inner passenger bracket—B			X		0.30	\$0.76	0.017	0	0.00	\$0.00	0.000	0	0.000
Lower bracket 1	X	X	X		0.25	\$0.46	0.031	1	0.25	\$0.46	0.031	1	0.031
Lower bracket 2—A	X	X			0.07	\$0.35	0.021	0.5	0.04	\$0.17	0.010	0.3	0.006
Lower bracket 2—B			X		0.07	\$0.47	0.013	0	0.00	\$0.00	0.000	0	0.000
Lower bracket 3	X	X	X		0.07	\$0.31	0.028	1	0.07	\$0.31	0.028	1	0.028
Lower bracket 4—A	X				0.07	\$0.39	0.017	0	0.00	\$0.00	0.000	0	0.000
Lower bracket 4—B		X			0.07	\$0.72	0.010	0	0.00	\$0.00	0.000	0	0.000

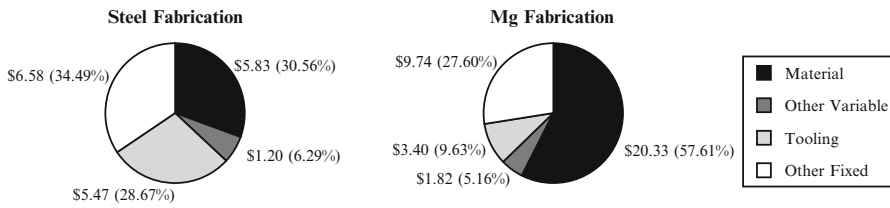
Lower bracket 4—C	X	50	0.07	\$0.47	0.013	0	0.00	\$0.00	0.000	0	0.000
Lower steering column support	X	150	0.27	\$0.47	0.032	1	0.27	\$0.47	0.032	1	0.032
Lower structure 1	X	150	0.40	\$1.66	0.000	1	0.40	\$1.66	0.000	1	0.000
Lower structure 2	X	150	0.25	\$0.97	0.000	1	0.25	\$0.97	0.000	1	0.000
Outer drivers side bracket	X	150	0.27	\$0.47	0.032	1	0.27	\$0.47	0.032	1	0.032
Reinforcement 1	X	150	0.30	\$0.49	0.032	1	0.30	\$0.49	0.032	1	0.032
Reinforcement 2	X	150	0.30	\$0.49	0.032	1	0.30	\$0.49	0.032	1	0.032
Upper bracket 1—A	X	100	0.75	\$0.88	0.028	0.5	0.38	\$0.44	0.014	0.3	0.008
Upper bracket 1—B	X	50	0.75	\$1.10	0.020	0	0.00	\$0.00	0.000	0	0.000
Upper bracket 2	X	150	0.07	\$0.31	0.014	1	0.07	\$0.31	0.014	1	0.014
Upper bracket 3	X	150	0.07	\$0.31	0.028	1	0.07	\$0.31	0.028	1	0.028
Upper bracket 4—A	X	100	0.10	\$0.38	0.028	0.5	0.05	\$0.19	0.014	0.3	0.008
Upper bracket 4—B	X	50	0.10	\$0.52	0.021	0	0.00	\$0.00	0.000	0	0.000
Upper bracket 5	X	150	0.75	\$0.75	0.035	1	0.75	\$0.75	0.035	1	0.035
Upper structure—A	X	100	1.97	\$3.22	0.067	0.5	0.98	\$1.61	0.034	0.3	0.020
Upper structure—B	X	50	1.79	\$2.90	0.033	0	0.00	\$0.00	0.000	0	0.000
Wiring harness bracket 1	X	150	0.07	\$0.31	0.028	1	0.07	\$0.31	0.028	1	0.028
Wiring harness bracket 2	X	150	0.07	\$0.31	0.028	1	0.07	\$0.31	0.028	1	0.028
Wiring harness bracket 3	X	150	0.20	\$0.41	0.030	1	0.20	\$0.41	0.030	1	0.030

(continued)

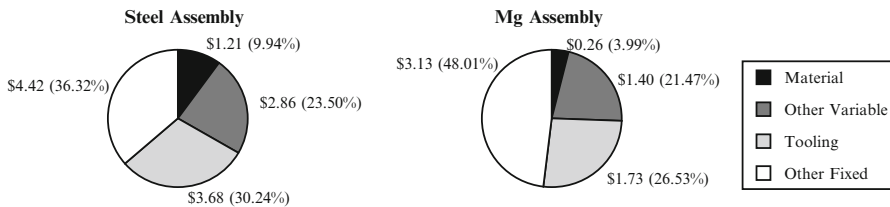




**Fig. 19.2** Projected standalone development costs for variant 1 of the magnesium and steel instrument panel at 75,000 units per year



**Fig. 19.3** Projected standalone fabrication costs for variant 1 of the magnesium and steel instrument panel at 75,000 units per year



**Fig. 19.4** Projected standalone assembly costs for variant 1 of the magnesium and steel instrument panel at 75,000 units per year

family are shown in Table 19.9. In the three variant steel design product family, the mass- and piece cost-weighted commonality metrics are less than that of the simple piece commonality metric. This is a result of the large number of brackets that are shared which have low masses and piece costs. The relative fabrication investment required for these brackets is higher; this is captured in the fabrication investment-weighted metric which is greater than the previous simple piece-, mass-, and piece cost-weighted metrics. The production volume-weighted commonality metric is lower than the simple piece metric due to the majority of product family commonality being between the variants one and two (the low production volume variant). The investment- and production volume-weighted metric captures both effects and is approximately the average of the two single metrics.

In the case of the three variant magnesium design product family, the simple piece commonality metric did not differ significantly from the mass or piece cost-weighted metrics. The investment-weighted commonality metric for this case is

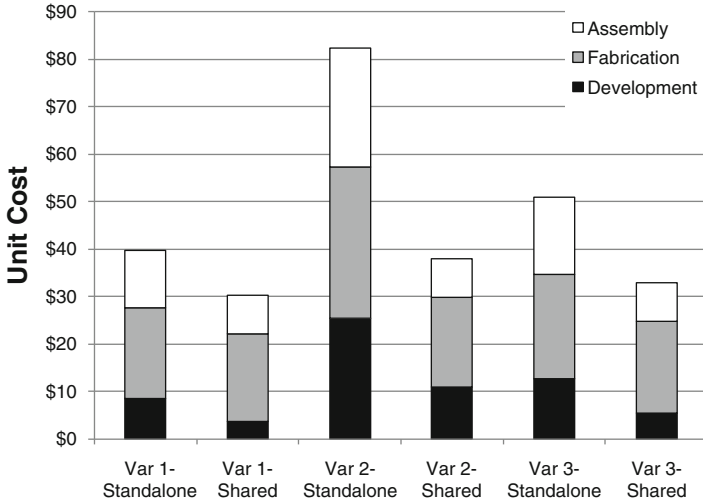


Fig. 19.5 Projected shared and standalone costs for three steel IP beam variants

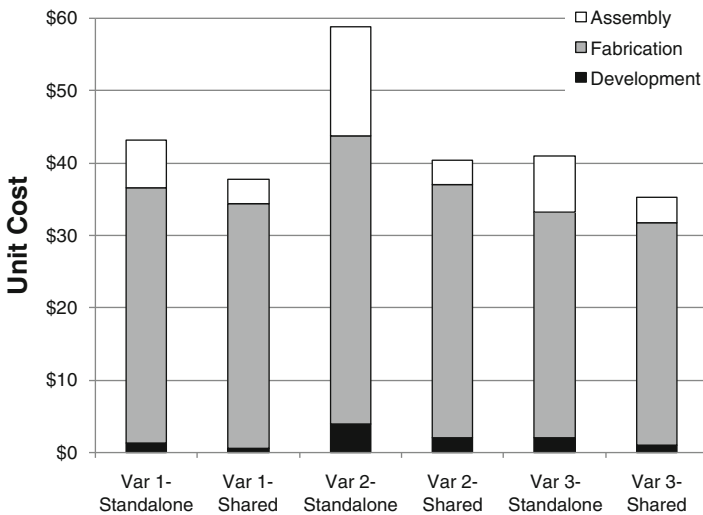


Fig. 19.6 Projected shared and standalone costs for three magnesium IP beam variants

almost 50 % higher than that of the simple piece metric. This is due to the significant investment associated with the die-cast structures shared by variants one and two. The production volume-weighted commonality metric is approximately a third less than that of the simple piece commonality metric. The investment/production volume metric combines the two but is closer to that of the production volume commonality metric. This metric’s calculation penalizes the

**Table 19.9** Commonality metrics and cost savings for IP beam product families

	Steel variants 1,2, and 3	Steel variants 1 and 2	Steel variants 1 and 3	Steel variants 2 and 3	Mg variants 1, 2, and 3	Mg variants 1 and 2	Mg variants 1 and 3	Mg variants 2 and 3
Piece commonality— $C_{Piece}$	0.65	0.94	0.58	0.58	0.25	0.20	0.50	0.00
Mass commonality— $C_{\phi=Mass}$	0.58	0.99	0.44	0.44	0.27	0.92	0.04	0.00
Piece cost commonality— $C_{\phi=Cost}$	0.55	0.94	0.40	0.40	0.26	0.89	0.04	0.00
Investment commonality— $C_{\phi=Invest}$	0.71	0.96	0.62	0.61	0.34	0.90	0.08	0.00
Production volume commonality— $C_{PV}$	0.62	0.94	0.58	0.58	0.18	0.20	0.50	0.00
Investment/production volume commonality— $C_{PV/\phi=Invest}$	0.67	0.96	0.62	0.61	0.21	0.90	0.08	0.00
Total savings	41 %	38 %	29 %	35 %	21 %	19 %	10 %	14 %
Fixed cost savings	49 %	45 %	35 %	40 %	36 %	33 %	19 %	23 %

magnesium design product family for sharing a high investment component with a variant that is produced at a low production volume.

The steel product family containing only variants one and two has similar values for all commonality metrics. For the other two product families, the mass- and piece-weighted metrics are lower due to the sharing of brackets which have lower masses and piece costs (as was the case for the three variant product family). It should be noted for the cases with two product variants, the production volume-weighted sharing metric collapses to the simple piece sharing metric. The total costs savings resulting from component sharing range from 29 % to 38 %.

The two variant product families for the magnesium design demonstrate the effects of alternative component sharing strategies. For the product family containing variants one and two, the simple piece sharing is significantly lower than all the other metrics (save the production volume sharing metrics to which it collapses in this case). While this product family shares only one component, it shares the largest component when weighted by mass, piece cost, and fabrication investment. This is reflected in the weighted commonality metrics. The product family containing variants one and three has a simple piece commonality metric value of 0.5. The weighted metrics range from 0.04 to 0.15, the 0.15 value being for the fabrication investment-weighted metric; this again is due to the significant investment required to fabricate brackets. The total savings range from 10 % to 19 % for these product families. Notably, the product family with no component sharing benefits from shared assembly processes (e.g., shared conveyance, robotics, and facilities). Even when the commonality metric is zero, the gross similarity of these products allows them to be produced on the same assembly line (albeit using different tools). The cost savings produced by the shared assembly processes are positive and significant because the two standalone variants are produced at relatively low production volumes.

The correlations between the various commonality metrics are shown in Table 19.10. The two-tailed significance is shown under the correlation coefficient. The piece commonality metric is not significantly correlated with any of the weighted metrics. It only correlates significantly with the production volume metric; this is to be expected given that they are equal for all but the two three variant product families (this is also true of the investment, investment/production volume commonality metrics). The mass commonality metric correlates well with both the piece cost and the investment commonality metrics. As seen in Table 19.7 and 19.8, mass, piece cost, and investment tend to correlate; as such, their commonality metrics would be correlated.

For the purpose of simple linear regression analysis, the production volume commonality and investment and production volume commonality metrics were excluded for the two variant product families because for the majority of product family cases (all those containing only two variants), they match exactly the values for the simple piece- and investment-weighted metrics, respectively. Table 19.11 shows the  $R^2$ ,  $F$  statistic, and significance for each correlation. The  $R^2$  for the piece commonality and investment commonality metrics are 0.54 and 0.53, respectively (both are significant at the 0.05 level). These metrics performed significantly better



**Table 19.10** Commonality metric correlations

	Mass commonality	Piece cost commonality	Investment commonality	Production volume commonality	Investment/ production volume commonality
Piece commonality— $C_{Piece}$	0.511 (0.196)	0.491 (0.217)	0.583 (0.130)	0.997* (0.000)	0.593 (0.121)
Mass commonality— $C_{\phi=Mass}$		0.999* (0.000)	0.974* (0.000)	0.514 (0.193)	0.969* (0.000)
Piece cost commonality— $C_{\phi=Cost}$			0.966* (0.000)	0.493 (0.214)	0.961* (0.000)
Investment commonality— $C_{\phi=Invest}$				0.581 (0.131)	0.992* (0.000)
Production volume commonality— $C_{PV}$					0.602 (0.114)

\*Correlation is significant at 0.01 level (two-tailed); significance shown in parentheses under correlation

**Table 19.11** Regression statistics for commonality metrics and total cost savings

	$R^2$	F-stat	Significance
Piece commonality	0.538	6.98	0.038
Mass commonality	0.370	3.53	0.109
Piece cost commonality	0.346	3.17	0.125
Investment commonality	0.526	6.65	0.042

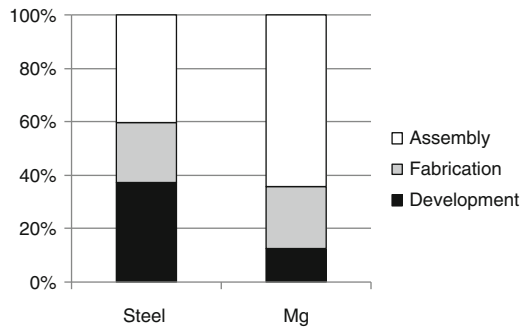
than the mass- or piece cost-weighted metrics in their relationship to resultant cost savings. The underperformance of the mass- or piece cost-weighted metrics matches well with the earlier observation that platform cost savings derived primarily from changes in assembly and development costs. Both of these costs are more closely tied to part count or complexity rather than mass or the piece costs that it drives.

Several authors have indicated that the key cost benefit of product families would derive from a reduction in fixed costs not, generally, from any reduction in piece costs (Robertson and Ulrich 1998). In light of this, the authors explored the effectiveness of the component sharing metrics in projecting fixed costs alone. As will be discussed further, subsequently, doing this may help to resolve the implications of component sharing even in the face of the potentially significant variation in piece cost that can exist across different materials and production technologies. The correlation between commonality metrics and fixed costs is presented in Table 19.12. For the purposes of this analysis, fixed costs include the

**Table 19.12** Regression statistics for commonality metrics and fixed cost savings

	$R^2$	$F$ -stat	Significance
Piece commonality	0.377	3.64	0.105
Mass commonality	0.483	5.61	0.056
Piece cost commonality	0.468	5.27	0.061
Investment commonality	0.617	9.65	0.021

**Fig. 19.7** Projected cost category savings as a portion of total cost savings



tooling, equipment, building, and overhead and maintenance costs related to manufacturing and development costs (since they are an upfront expenditure, not dependent on production volume). When assessing the relationships with fixed costs, the mass-weighted and piece cost-weighted measures have higher  $R^2$  values than the simple piece commonality metric. Of the metrics investigated, the fabrication investment-weighted metric has the highest  $R^2$  value and is the only metric significant at the 0.025 level. While this may seem like an expected result, as noted above and seen in Fig. 19.7, the majority of modeled cost savings from component sharing derived from reduced assembly and development costs. As such, this result would indicate that the fabrication investment-weighted metric may provide even more predictive ability than would otherwise be expected.

### 19.5 Discussions

One way of combating the effects of the increased product variety and decreasing product lifetimes has been the introduction of product families. For these families to be effective, they must be organized in a way that achieves the goals of the firm. A number of authors have suggested that effective design performance measures should promote product family designs that support those goals. This work has examined the fidelity of several possible early-stage commonality measures with the goal of reducing the overall (fabrication, assembly, and development) costs of the product. Specifically, through the use of a case study, several metrics were assessed to determine their relationship with the cost savings associated with specific component sharing strategies. To obtain consistent costs, the method of process-based cost modeling was used to project costs for families of products both

when produced with potential component sharing (shared cost) and in a context where all parts are produced independently (standalone cost).

Four metrics of component commonality were compared to total and fixed cost savings that were a result of component sharing. When considering total cost savings, both the mass- and piece cost-weighted commonality metrics performed poorly in relating component sharing to benefit. In contrast, the simple piece commonality metric and the fabrication investment-weighted metric performed considerably better in their correlation to total cost savings. Notably, the product families analyzed included components that had relatively large masses and piece costs and, therefore, did not benefit greatly from component sharing. This is most evident in the case of the magnesium IP beam's main structure, for which the costs derived from variable costs in general, and material cost in particular, limited the amount of savings that could be derived from component sharing. Although some might view this as a special characteristic of the specific case of study, it does serve to highlight the distortion that is possible from a mass or piece cost weighting scheme.

To accommodate the significant variation in piece costs across the technologies under study, the effectiveness of the various metrics was also assessed against fixed costs savings alone. In this comparison, the simple piece commonality metric performed poorly. The mass- and piece cost-weighted commonality metrics performed somewhat better. The fabrication investment-weighted commonality metric performed well with the highest-observed  $R^2$  of 0.62 and significance at the 0.025 level. At first, this may seem like an obvious finding: sharing parts with the most fabrication investment provides the most opportunity for cost savings. However, the modeled cost results indicate that the majority of savings derived from component sharing are in the development and assembly cost categories, not fabrication. These results suggest that a metric based on fabrication investment savings may have explanatory value for other forms of fixed cost savings.

When developing product families, it is important to consider which types of components are being shared and what type of product is being produced. The exploration of metrics presented in this work suggest that assessing alternative product family concepts based on their component sharing can provide insight on the potential economic consequence. Furthermore, certain component sharing metrics seem to provide more fidelity in projecting ultimate cost savings. In particular, our results suggest that given access either to process-based cost models of fabrication or to accounting data of sufficiently similar parts, various product family strategies can be assessed with significantly improved insight using a fabrication investment-weighted metric, as presented herein.

Such a metric should also make it possible to enhance the development of variant products derived from those already in the market or under development. Ranking component sharing targets according to fabrication investment would allow designers to readily identify those components which would be the most valuable to share; this should allow for a more targeted and effective component sharing strategy. This metric also seems the most robust for those product families with a significant percentage of variable costs. Even for such cases, the fabrication investment weight metric should serve as an effective proxy to identify which product family concepts would capture the greatest percentage of available cost savings.

In the end, our results suggest that product design managers would benefit from encouraging their design teams to frequently evaluate platform design projects in terms of component sharing. For high-volume production environments, a fabrication investment-weighted commonality metric like the one presented here appears to be particularly effective. Implementing even a simple piece count metric will require some standardized criteria by which designers can evaluate whether a component is considered common. In the authors' experience, designers are reasonably effective at assessing the criteria applied herein—the need for a separate primary forming tool. In contexts where that is not possible, managers should consider other approaches, such as that outlined by Kota et al. (2000).

To realize the increased diagnostic value of an investment-weighted metric, managers would also need to establish the process by which fabrication investments are estimated (and by which these estimations are kept up-to-date). As discussed in the literature review section, process-based cost modeling as well as several parametric modeling approaches have proven broadly useful for estimating investments based on project characteristics available during early design. In the authors' experience such models must be periodically updated, but for many industries that need occur only every 1–2 years. Over that period, dozens, if not hundreds, of design projects can benefit from the information provided by the model. For most firms, that scale of benefit easily justifies the data collection and analysis effort required to realize and maintain such a model.

It is reasonable to question whether some firms, with sufficient investment in cost modeling capability, could simply employ such cost models to directly (and more completely) estimate the cost consequences of a platforming project rather than relying on a proxy metric like those investigated. The authors suspect that such contexts exist and that individual firms should consider this possibility. Nevertheless, there are clearly cases where proxy metrics are appropriate, primarily to accommodate the state of knowledge of a given design team. Most notably, a fabrication investment proxy like that explored here does not require knowledge of integration and assembly issues that naturally can only be known once a complete set of components are designed. Similarly, fabrication weighting obviates the need to know information about drivers of variable costs (materials source, labor, or energy) which tend to be much more regionally variable than their investment analogs. In both cases, the use of the proxy metric clearly limits the amount of additional information burden on the design team but still provides significant explanatory information about the ultimate cost savings realizable through effective platforming.

## 19.6 Limitations and Future Work

This work used process-based cost models of development, fabrication, and assembly. The designs were assumed to be a posteriori (or bottom-up) designed product families, with each variant derived from the preceding one. This work did not

include the additional cost associated with a priori (or top-down) platform development (Maier and Fadel 2001). Based on input from the firms from which case data was collected, this work focused on the reduction in fixed costs afforded by component sharing. Process-based cost models were used to project these reductions. Other authors have suggested that product platforming can reduce variable costs as well (Krishnan and Gupta 2001), but based on input from case firms, these effects were viewed to be minor and were not included in this analysis. This may not hold for all industries and should become a focus of future study.

This work also did not take into account the operational or market effects of commonality or the related characteristic variety; other researchers have noted the difficulty in calculating such costs (e.g., Zhang and Tseng 2007). This work assumes that all product variants are necessary to meet market needs (given the requirement for different IP beam sizes) and that the benefits of providing for these needs greatly outweigh the costs of the associated variant level variety. Revenue implications have not been considered here. Reduction in consumer-perceived product variety due to the use of common components can impact sales and pricing (Clark and Fujimoto 1989; Thomas 1992; Nelson et al. 2001; Oshri and Newell 2005; Krishnan and Gupta 2001; Gonzalez-Zugasti et al. 2000; Yu et al. 1999; Kim and Chhajed 2001). Such revenue effects can easily outweigh cost benefits and should be considered carefully. It is not clear, however, that any of the metrics explored here could resolve the revenue effects of a given commonality decision. The effects of commonality (the opposite of variety) on inventory costs have been the focus of much commonality/variety research (Fixson 2007). Inclusion of inventory carrying costs might enhance the performance of the piece cost commonality metric given that piece cost is a major driver of inventory holding costs (Waters 1992). Another benefit of the reduction in variety is the reduced need for training and setup (Fixson 2006). The inclusion of these benefits might enhance the performance of the piece commonality metric (assuming constant training and setup needs) or the investment commonality metric (assuming training and setup are correlated with investment). Future work would benefit from the inclusion of a detailed analysis to assess the costs of variety at the product variant (assembly) level as well as the effects of the reduction in variety (at the component level) on overall product family costs.

Finally, this work contains a limited number of cases and deals mainly with metal-forming processes. This focused analysis was able to highlight some promising trends but is not statistically conclusive. Future work should expand these cases to include a larger number of variants at a wider range of production volumes as well as alternative manufacturing processes. Along these lines, it would be interesting to investigate cases where cost savings were dominated by other cost categories. Notably, here platform-derived savings were mostly associated with assembly and development. A context with relatively low part count (i.e., to reduce assembly and development costs) and relatively high tooling investment (i.e., to raise part production costs) would represent an important test case. A greater number of cases would allow the two metrics that were not assessed in this work (production volume-weighted metric and the fabrication investment/production

volume metric) to be analyzed. There is no reason to limit the goal of the product family to reducing costs; these methods could also be used to limit the different types of materials used. This could allow for the reclamation of certain types of materials to become economically feasible and thus increase recycling and reuse. The cost differential between a “recycling enhanced” and a baseline case could be used along with the costs/benefits of recycling to determine a suitable strategy. Other goals for the metrics could include distribution-related quantities such as shipping miles or number of supplies.

## References

- Alizon F, Khadke K, Thevenot HJ, Gershenson JK, Marion TJ, Shooter SB, Simpson TW (2007) Frameworks for product family design and development. *Concurr Eng Res Appl* 15(2): 187–199
- Angelis DI, Lee C-Y (1996) Strategic investment analysis using activity based costing concepts and analytical hierarchy process techniques. *Int J Prod Res* 34(5):1331–1345
- Blecker T, Abdelkafi N (2007) The development of a component commonality metric for mass customization. *IEEE Trans Eng Manage* 54(1):70–85
- Cirincione RJ (2008) A study of optimal automotive materials choice given market and regulatory uncertainty. Thesis (S.M.), Massachusetts Institute of Technology, Cambridge
- Clark KB, Fujimoto T (1989) Lead time in automobile product development explaining the Japanese advantage. *J Eng Technol Manage* 6(1):25–56
- Cleland K (2001) *Basic extinctions. Financial management*. Seven Publishing Group
- Collier DA (1981) The measurement and operating benefits of component part commonality. *Decision Sci* 12(1):85–96
- Cooper R, Kaplan RS (1988) Measure costs right: make the right decisions. *Harv Bus Rev* 66(5): 96–103
- Eisenhardt KM (1989) Building theories from case study research. *Acad Manage Rev* 14(4): 532–550
- Field F, Kirchain R, Roth R (2007) Process cost modeling: strategic engineering and economic evaluation of materials technologies. *J Oper Manage* 59(10):21–32
- Fixson SK (2005) Product architecture assessment: a tool to link product, process, and supply chain design decisions. *J Oper Manage* 23(3–4):345–369
- Fixson S (2006) Effective product platform planning in the front end. In: Simpson TW, Siddique Z, Jianxin J (eds) *Product platform and product family design*. Springer, New York, pp 305–333
- Fixson SK (2007) Modularity and commonality research: past developments and future opportunities. *Concurr Eng Res Appl* 15(2):85–111
- Fuchs ERH, Field FR, Roth R, Kirchain RE (2008) Strategic materials selection in the automobile body: economic opportunities for polymer composite design. *Compos Sci Technol* 68(9): 1989–2002
- Gonzalez-Zugasti JP, Otto KN, Baker JD (2000) A method for architecting product platforms. *Res Eng Des* 12(2):61–72
- Guerrero HH (1985) The effect of various production strategies on product structures with commonality. *J Oper Manage* 5(4):395–410
- Gupta AK, Souder WE (1998) Key drivers of reduced cycle time. *Res Technol Manage* 41(4): 38–43
- Hillier MS (2002) The costs and benefits of commonality in assemble-to-order systems with a (Q, r)-policy for component replenishment. *Eur J Oper Res* 141(3):570–586

- Ho C-J, Li J (1997) Progressive engineering changes in multi-level product structures. *Omega* 25(5):585–594
- Jiao J, Tseng MM (2000) Understanding product family for mass customization by developing commonality indices. *J Eng Des* 11(3):225–243
- Johnson M, Kirchain R (2009a) Quantifying the effects of parts consolidation and development costs on material selection decisions: a process-based costing approach. *Int J Prod Econ* 119(1):174–186. doi:[10.1016/j.ijpe.2009.02.003](https://doi.org/10.1016/j.ijpe.2009.02.003)
- Johnson MD, Kirchain RE (2009b) Quantifying the effects of product family decisions on material selection: a process-based costing approach. *Int J Prod Econ* 120(2):653–668
- Kar AM (2007) A cost modeling approach using learning curves to study the evolution of technology. Thesis (S.M.), Massachusetts Institute of Technology, Cambridge, MA
- Kim K, Chhajed D (2001) An experimental investigation of valuation change due to commonality in vertical product line extension. *J Prod Innov Manage* 18(4):219–230
- Kirchain R, Field FR (2001) Process-based cost modeling: understanding the economics of technical decisions. In: Buschow KHJ, Cahn RW, Flemings MC, Ilschner B, Kramer EJ, Mahajan S (eds) *Encyclopedia of materials science & engineering*, vol 2. Elsevier Science, San Diego, CA, pp 1718–1727
- Kota S, Sethuraman K, Miller R (2000) A metric for evaluating design commonality in product families. *J Mech Des* 122(4):403–410
- Kranenburg AA, Van Houtum GJ (2007) Effect of commonality on spare parts provisioning costs for capital goods. *Int J Prod Econ* 108(1–2):221–227. doi:[10.1016/j.ijpe.2006.12.025](https://doi.org/10.1016/j.ijpe.2006.12.025)
- Krishnan V, Gupta S (2001) Appropriateness and impact of platform-based product development. *Manage Sci* 47(1):52–68
- Labro E (2004) The cost effects of component commonality: a literature review through a management-accounting lens. *Manuf Ser Oper Manage* 6(4):358–367
- MacDuffie JP, Sethuraman K, Fisher ML (1996) Product variety and manufacturing performance: evidence from the international automotive assembly plant study. *Manage Sci* 42(3):350–369
- Maier JRA, Fadel GM (2001) Strategic decisions in the early stages of product family design. Paper presented at the design engineering technical conference, Pittsburgh, PA, 9–12 Sept 2001
- Martin MV, Ishii K (1996) Design for variety: a methodology for understanding the costs of product proliferation. Paper presented at the ASME design engineering technical conference, Irvine, CA, 18–22 Aug 1996
- Martin MV, Ishii K (1997) Design for variety: development of complexity indices and design charts. Paper presented at the ASME design engineering technical conference, Sacramento, CA, 14–17 Sept 1997
- Maskell BH (1991) Measurement of production flexibility. In: *Performance measurement for world class manufacturing: a model for American companies*. Productivity, Cambridge, MA, pp 171–202
- McDermott CM, Stock GN (1994) The use of common parts and designs in high tech industries: a strategic approach. *Prod Inv Manage J* 35(3):65–69
- Meyer MH, Lehnerd AP (1997) The power of product platforms. The Free, New York
- Moscato DR (1976) The application of the entropy measure to the analysis of part commonality in a product line. *Int J Prod Res* 14(3):401–406
- Muffatto M (1999) Introducing a platform strategy in product development. *Int J Prod Econ* 60–61:145–153
- Nelson SA, Parkinson MB, Papalambros PY (2001) Multicriteria optimization in product platform design. *J Mech Des* 123(2):199–204
- Nobelius D, Sundgren N (2002) Managerial issues in parts sharing among product development projects: a case study. *J Eng Technol Manage* 19(1):59–73
- Noreen E (1991) Conditions under which activity-based cost systems provide relevant costs. *J Manage Accounting Res* 3(Fall):159–168
- Noreen E, Soderstrom N (1994) Are overhead costs strictly proportional to activity? *J Accounting Econ* 17(1–2):255–278

- Oshri I, Newell S (2005) Component sharing in complex products and systems: challenges, solutions, and practical implications. *IEEE Trans Eng Manage* 52(2):509–521
- Park J, Simpson TW (2008) Toward an activity-based costing system for product families and product platforms in the early stages of development. *Int J Prod Res* 46(1):99–130
- Pine BJ (1991) Paradigm shift: from mass production to mass customization. Thesis (S.M.), Massachusetts Institute of Technology, Cambridge, MA
- Pine BJ (1993) Mass customization: the new frontier in business competition. Harvard Business School Press, Boston
- Qian L, Ben-Arieh D (2008) Parametric cost estimation based on activity-based costing: a case study for design and development of rotational parts. *Int J Prod Econ* 113(2):805–818
- Robertson D, Ulrich K (1998) Planning for product platforms. *Sloan Manage Rev* 39(4):19–31
- Rosenthal SR, Tatikonda MV (1992) Time management in new product development: case study findings. *J Manuf Syst* 11(5):359–368
- Roth R, Shaw J (2002) Achieving an affordable low emission steel vehicle; an economic assessment of ULSAB-AVC program design. Paper presented at the SAE 2002 world congress, Detroit, MI
- Sanderson S, Uzumeri M (1995) Managing product families: the case of the Sony Walkman. *Res Policy* 24(5):761–782
- Schonberger R (1987) Frugal manufacturing. *Harvard Bus Rev* 65(5):95–100
- Simpson TW, Marion T, De Weck OL, Holtta-Otto K, Kokkolaras M, Shooter SB (2006) Platform-based design and development: current trends and needs in industry. Paper presented at the ASME international design engineering technical conference, Philadelphia, PA, USA, 10–13 Sept 2006
- Skold M, Karlsson C (2007) Multibranded platform development: a corporate strategy with multimanual challenges. *J Prod Innov Manage* 24(6):554–566
- Thevenot HJ, Simpson TW (2006) A comprehensive metric for evaluating component commonality in a product family. Paper presented at the ASME 2006 international design engineering technical conferences & computers and information in engineering conferences, Philadelphia, PA, 10–13 Sept 2006
- Thomas LD (1992) Functional implications of component commonality in operational systems. *IEEE Trans Syst Man Cybern* 22(3):548–551
- Thonemann UW, Brandeau ML (2000) Optimal commonality in component design. *Oper Res* 48(1):1–19
- Tsubone H, Matsuura H, Satoh S (1994) Component part commonality and process flexibility effects on manufacturing performance. *Int J Prod Res* 32(10):2479–2493
- Tu YL, Xie SQ, Fung RYK (2007) Product development cost estimation in mass customization. *IEEE Trans Eng Manage* 54(1):29–40
- Ulrich K (1995) The role of product architecture in the manufacturing firm. *Res Policy* 24:419–440
- Ulrich KT, Pearson S (1998) Assessing the importance of design through product archaeology. *Manage Sci* 44(3):352–369
- Uzumeri M, Sanderson S (1995) A framework for model and product family competition. *Res Policy* 24(4):583–607
- Vakharia AJ, Parmenter DA, Sanchez SM (1996) The operating impact of parts commonality. *J Oper Manage* 14:3–18
- von Braun CF (1990) The acceleration trap. *Sloan Manage Rev* 32(1):49–58
- Wacker JG, Treleven M (1986) Component part standardization: an analysis of commonality sources and indices. *J Oper Manage* 6(2):219–244
- Waters DC (1992) Inventory control and management. Wiley, Chichester
- Yu JS, Gonzalez-Zugasti JP, Otto KN (1999) Product architecture definition based upon customer demands. *J Mech Des* 121:329–335
- Zacharias NA, Yassine AA (2008) Optimal platform investment for product family design. *J Intell Manuf* 19(2):131–148
- Zhang M, Tseng MM (2007) A product and process modeling based approach to study cost implications of product variety in mass customization. *IEEE Trans Eng Manage* 54(1):130–144



# Chapter 20

## Managing Design Processes of Product Families by Modularization and Simulation

Qianli Xu and Roger J. Jiao

**Abstract** Managing multiple design projects of product family design necessitates exploitation of commonalities among various variant projects to achieve reduced time and cost. It is important to establish a design process architecture that captures the relevant design process information and to model the design process so as to integrate the logic of design process planning. This chapter describes a design process modularization approach to establish the design process architecture and an integrated modeling and simulation method based on Petri nets (PNs). This framework adopts a generic variety structure of representing diverse variant design processes. A modular design project planning architecture is structured by identifying design process modules using a fuzzy clustering approach. Based on the modular design process, a timed colored Petri net model is formulated to integrate the planning logic of design process configuration, while evaluating design project performance through simulation. Application of the proposed framework in a car dashboard product family design demonstrates promising results of design process management based on modularization and PN simulation.

---

This book chapter is compiled from the authors' prior publications in ASME Journal of Mechanical Design (DOI: [10.1115/1.3149844](https://doi.org/10.1115/1.3149844) & DOI: [10.1115/1.3125203](https://doi.org/10.1115/1.3125203)) (©ASME 2008), reprinted with permission.

Q. Xu

Agency for Science, Technology and Research, Institute for Infocomm Research, Singapore, Singapore

R.J. Jiao (✉)

Georgia Institute of Technology, Woodruff School of Mechanical Engineering, Atlanta, GA 30332-0405, USA  
e-mail: [rjiao@gatech.edu](mailto:rjiao@gatech.edu)

## 20.1 Introduction

Manufacturing enterprises typically manage multiple design projects where a few product variants are developed in parallel according to customers’ requirements. These design projects are rarely managed in isolation; rather they share common technical information (e.g., product/process platforms), facilities (e.g., materials and equipment), budgets, and human resources. It is essential to synchronize the process of multiple design projects and reuse common resources so as to fully exploit the benefits of platform-based product development. To do so, manufactures need to deal with complicated task planning that involves a number of design activities and resources related to the multiple product variants development (Meier et al. 2007; Jiang et al. 2008).

This chapter focuses on the management of design process of product variants, where a design process refers to a sequence of interrelated activities that transform customer needs (CNs) to a complete description of products and/or production plans (Kusiak and Wang 1993a, b). Similar to the domain mapping concept in axiomatic design (Suh 1990), the process of product variant design can be roughly divided into three stages as shown in Fig. 20.1. By employing a platform-based rationale, each stage is supported by predefined platforms, such as product portfolio, product platform, and process platform. For example, the requirement analysis stage deals with the transformation of CNs into functional requirement (FRs). This stage can be further decomposed into a set of design activities, including analyzing customer orders, dispatching tasks, checking compatibility, and issuing design specifications. The product planning stage deals with the synthesis of product configurations that provide the product features. Typical design activities in this stage include selecting and designing components, quality assessment, validating product with customers, and designing components, quality assessment, validating product with customers,

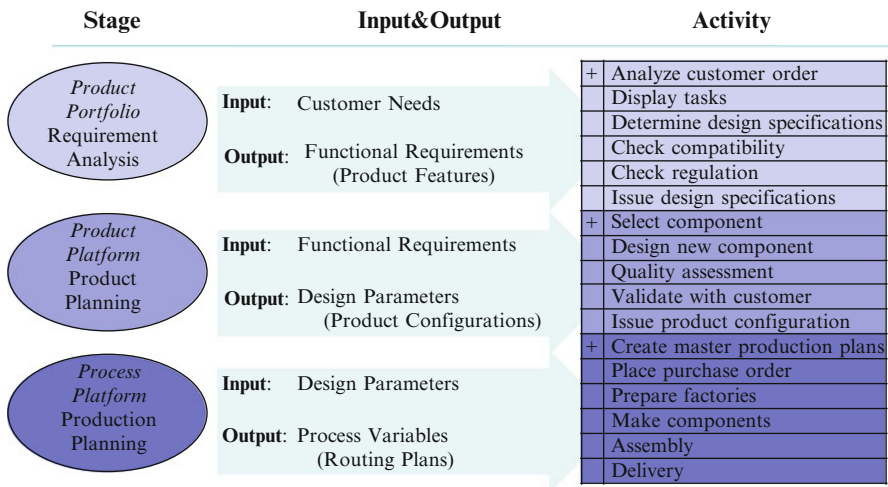


Fig. 20.1 A general design process of product variant

and issuing product configurations. In the production planning stage, the production process variables are finalized as routing plans. This stage involves activities such as creating master production plans, selecting suppliers, preparing factories, making components, assembling, and delivering. Depending on the nature of the design problem, a product variant design process may involve all or part of the above stages.

The process perspective of product family design gives rise to two fundamental issues, namely, (1) how to establish a design process architecture that captures the relevant design process information and (2) how to model the design process so as to integrate the reasoning logic of process plans and facilitate automated decision-making. A design process architecture provides a means to capturing the characteristics of design processes. Design information can be embedded in the architecture for reuse in future designs. Considering that the product development strategy adopted by a company is relatively stable, the product family design processes may exploit the similarities and commonalities among the design projects, which are considered as variant design projects. Such multiple design project variants assume a number of possibilities of reusing design teams, activities sequences, and resources (Chen and Li 2003; Yassine and Braha 2003).

For design process modeling, the objectives are (1) to capture the semantics associated with the product variant design process and (2) to provide decision support in the subsequent planning stage. In essence, a formal and comprehensive modeling scheme is needed to coordinate the diversified, distributed activities associated with the design process (Park and Cutkosky 1999). Moreover, a formal design process model enhances analytical rigor, in the sense that structured design process data can facilitate the application of computational techniques for reusing design process information (Ross 1977).

To tackle these issues, this chapter proposes (1) a design process modularization approach to establish the design process architecture and (2) an integrated modeling and simulation method based on Petri nets (PNs). This research adopts the generic variety structure (GVS) for representing the product variant design processes. Accordingly, a modular architecture is build where design process modules are identified using the fuzzy clustering approach. Based on the modular design process, a timed colored Petri net (TCPN) model is used to integrate the reasoning logic of process configuration and evaluate the project performance through simulation. The methods are applied in the design process management for automobile products.

## 20.2 Background Review

This section gives an overview of methods and tools for representing the design processes and developing process models for project configuration. In particular, various design process structures are examined and their roles in supporting process management are discussed. The establishment of modular design process structure is of particular interest. Next, a few design process models are discussed focusing on their roles in supporting process configurations.

### ***20.2.1 Generic Design Process Structure***

Product family design involves a dynamic and flexible process where multiple alternative roadmaps exist for developing product variants (Gonzalez-Zugasti et al. 2001). These roadmaps represent a series of project configurations depending on a number of unforeseen factors. Project planning is concerned with generating project configurations in terms of the design activity sequences and resource allocation. Subsequently, the configurations must be evaluated against certain objective functions, such as time and cost. Typically, a configuration design problem is characterized by the generation of solutions to satisfy a set of design requirements, based on “a fixed, predefined set of components, where a component is described by a set of properties and ports for connecting it to other components” (Mittal and Frayman 1989).

Alizon et al. (2007) categorize the product family design process into four types according to the development drivers (i.e., platform-driven and product-driven) and the availability of product information (i.e., top-down and bottom-up). While each type may involve design processes with distinctive objectives and information flows, the fundamental processes are similar for multiple design projects belonging to the same type. Thus, it is possible to develop a generic process structure shared by multiple design projects. A generic representation has been proven to be an effective means to describe a large number of variants with minimal data redundancy. For example, the generic bill-of-material (GBOM) defines a generic product as a set of variants that can be identified through specifying alternative values for a set of parameters (van Veen 1992; Hegge and Wortmann 1991). The Generic Bill-of-Material-and-Operation (GBOMO) is a data structure that unifies BOMs and routings to accommodate a large number of product and process variants (Jiao et al. 2000). These generic variety structures mainly focus on product and production process. A broader scope is needed for project management in the context of product families.

### ***20.2.2 Modular Design Projects***

The concept of modular design has profound influence in engineering design. Four types of modularity have been identified, namely, product modularity, process modularity, organization modularity, and innovation modularity (Fixson 2007). While the majority of reported work has focused on product modularity, the other types of modularity are being recognized as essential design problems (Upton and McAfee 2000; Fixson 2007). Leger and Morel (2001) propose a modular process that allows for breaking up the monitoring process of a part of a hydropower plant maintenance process into four subprocesses. Watanabe and Ane (2004) investigate the relationship between product modularity and process modularity. The results suggest that product modularity increases the processing flexibility of machines

and the agility of a manufacturing system. Balakrishnan and Brown (1996) study the commonality across products and its implications to the shared set of processing steps.

Another important issue in modular design is the identification of modules. In general, module identification is contingent on the representation schemes. In such a respect, modularity analysis methods have been developed based on design structure matrices (DSMs) and domain mapping matrices (DMMs), due to their elegant representation and support for mathematical operations. Typical DSM-based analytical methods include partitioning (Gebala and Eppinger 1991), clustering (Yassine and Braha 2003), banding, tearing, and sequencing (Browning 2002). A common objective of these methods is to group or reorder the DSM elements such that information exchange within groups is maximized while that between groups is minimized. Kusiak and Wang (1993a) propose a decomposition method to cluster activities involved in the design process. Kusiak and Wang (1993b) propose a triangulation algorithm which can generate a sequence of design activities such that the number of cycles is minimized. Pimmler and Eppinger (1994) propose the decomposition method to create architecture and teams based on numerical DSMs and the heuristic swapping algorithm. Fernandez (1998) proposes a clustering algorithm to find a trade-off between the costs of being inside a cluster and the overall system benefit. Sharman et al. (2002) use design syntax and semantics for the DSM representation. Yu et al. (2003) apply the DSM to identify highly interactive groups of product elements and cluster them into modules. Seol et al. (2007) develop a design process modularization method, where design processes are restructured based on modular synthesis. However, it remains to be addressed to model the design process of product families to deal with the process varieties and the intrinsic similarity among them.

### ***20.2.3 Design Process Models***

Prevalent design process models include PERT (Program Evaluation and Review Technique) (Wiest and Levy 1977), SADT (structured analysis and design technique) (Ross 1977), IDEF3 (Integrated DEFinition for Process Description Capture Method) (Mayer et al. 1995), WBS (work breakdown structure) (Liu and Horowitz 1989), and DSM (Steward 1981). However, they fall short to capture the rich interdependencies, priorities, and resource requirements of the design activities (Kumar and Ganesh 1998).

Petri nets have been recognized as a valuable tool for representing and analyzing dynamic, concurrent, asynchronous systems. Owing to the mathematical rigor and graphical modeling flexibility, PNs are capable of behavioral property analysis, performance evaluation, and discrete-event simulation based on the same model (Zurawski and Zhou 1994). Recently, there is an increasing number of PN applications in design process planning (Kumar and Ganesh 1998). Liu and Horowitz (1989) combine PNs with AND/OR graph to describe the project

breakdown structure and the specification of relationships between different project information types. Van der Aalst and Van Hee (1996) study the potential of PNs for constructing formal semantics, which facilitate a precise and unambiguous description of the behaviors of the business process. Raposo et al. (2000) develop a library of coordination mechanisms using PNs to enable management and reuse of workflow processes. Kao et al. (2006) propose a decision-making framework for project portfolio management, where high-level PNs are used to capture the workflow, and the temporal relationships of activities and resource types. Jiang et al. (2008) propose a timed colored PNs workflow model which features a process view for collaborative product development.

However, existing methods lack comprehensive representation schemes and decision-making techniques that account for the complexity and flexibility of design processes, especially when a large number of process variants are involved. Moreover, a generic design process structure is not yet developed based on PNs. Effective design process modeling is contingent on a clear understanding and proper generalization of the product variant design process.

## 20.3 Product Variant Design Process Structure

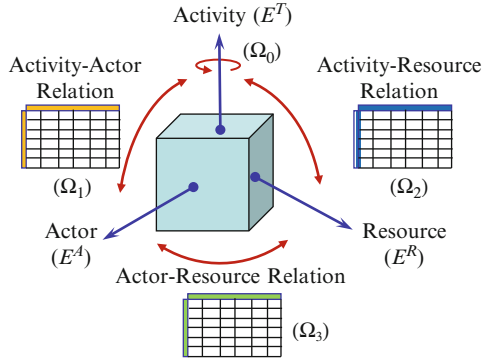
To establish a design process architecture, this section presents a fuzzy clustering method which facilitates the building of a GVS to capture the design process information.

### 20.3.1 Design Process of Product Variants

A design process ( $p$ ) of product variants is represented as a tuple:  $p \sim \langle E, \Omega \rangle$ , where  $E$  represents the basic elements of the process and  $\Omega$  represents the relationships between these elements. The elements ( $E$ ) of a process consist of activity/task ( $E^T$ ), actor ( $E^A$ ), and resource ( $E^R$ ), i.e.,  $E \sim \langle E^T, E^A, E^R \rangle$ . An activity is an instance of the task that is required by the process, uses up resources, and takes time to complete (Meredith and Mantel 2003). A task can be formally represented as  $E^T \sim \langle I^T, O^T \rangle$ , where  $I^T$  and  $O^T$  denote input and output messages, and a message is an abstraction of design information, such as CNs, FRs, design parameters (DPs), and process variables (PVs) (Suh 1990). An actor is a person or a team responsible for carrying out the design activity. A project usually requires the cooperation of different types of actors. A resource is a facility that is required to carry out the activities, such as equipment, materials, and budgets. A complete design project consists of a number of activities, actors, and resources.

The relationships between these elements are represented as a four-tuple:  $\Omega \sim \langle \Omega_0, \Omega_1, \Omega_2, \Omega_3 \rangle$ . The interdependencies and/or sequences of activities are

**Fig. 20.2** Viewing design processes from three perspectives

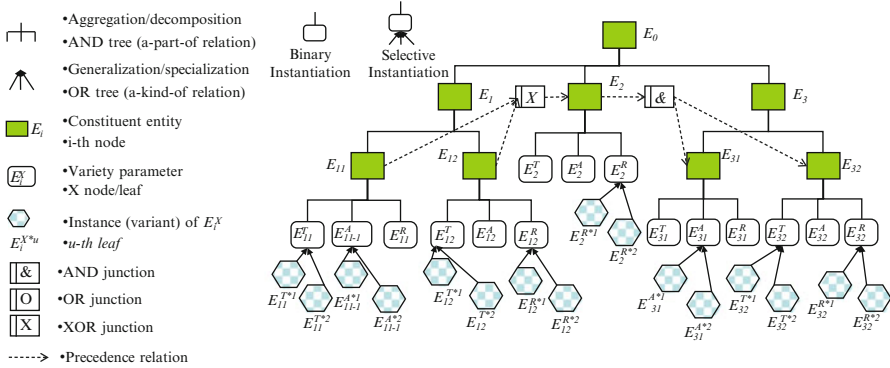


represented as  $\Omega_0 : E^T - E^T$ , i.e., whether an activity should be carried out before or after another activity. The correspondence between activities and actors is represented as  $\Omega_1 : E^T - E^A$ , i.e., who is the major actor for the activity. An actor may be responsible for multiple activities and each activity may involve one or more actors. Thus, there is a many-to-many mapping relation between  $E^T$  and  $E^A$ . Similarly, the correspondence between activities and resources is represented as  $\Omega_2 : E^T - E^R$ , i.e., what resources are required for carrying out an activity. A particular resource can be used in different activities, and an activity usually requires multiple types of resources. Therefore, the mapping relationship between  $E^T$  and  $E^R$  is many-to-many. Finally, the relationship between actors and resources is represented as  $\Omega_3 : E^A - E^R$ , which denotes an actor’s privilege or preference to the resources. The multiple elements and their relationships in a design process are illustrated in Fig. 20.2.

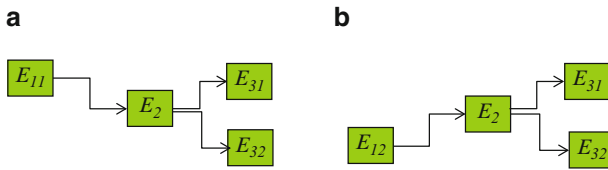
### 20.3.2 Generic Variety Structure

Based on the description of design process, a GVS is established as a hierarchical structure. A GVS is suitable to describe design process owing to the inherent similarities among multiple variant design processes. The characteristics of the GVS include four aspects: (1) design process hierarchy, (2) variety parameters, (3) generic item and indirect identification, and (4) configuration constraints. The readers may refer to Xu and Jiao (2010) for a discussion of GVS and its characteristics. Furthermore, the GVS adopts the concept of generic routing, which deals with the precedence links and junctions that show the logic of process branching. Figure 20.3 shows a GVS for representing design processes with precedence relations and junctions.

The leaf node activities are connected to each other using precedence links and junctions. These activities in combination with the sequence relations constitute a generic routing of process varieties. The precedence links are used to restrict the



**Fig. 20.3** A generic structure for representing process varieties



**Fig. 20.4** Routing variants derived from the generic routing. (a) Routing variant I. (b) Routing variant II

sequence of activities—an activity with a precedence link may not start until its preceding activity is completed. At the same time, a set of junctions is defined to represent the logic of process branching. The AND junction necessitates the parallel operation of a few activities in relation to a single activity. The OR junction shows the mutually compatible selective relation of a few activities, and the XOR junction shows the mutually exclusive selective relation of a few activities. Using the above example, two variants can be generated from the generic routing, as are shown in Fig. 20.4.

## 20.4 Project Module Identification

The GVS deals with three major project elements ( $E^T$ ,  $E^A$ , and  $E^R$ ), along with their relationships ( $\Omega_0, \Omega_1, \Omega_2$  and  $\Omega_3$ ). The relationships are denoted as DSMs and DMMs, which are used to identify module of activities (MoAs) based on three types of modularity. The module identification process is carried out using the fuzzy clustering method, which employs the representation and manipulation of the DSM and DMM.



**Fig. 20.5** DSM for module identification. (a) Original DSM. (b) Clustered DSM

	a	b	c	d	e	f	g
a	1	1			1		
b	1	1			1		
c			1				1
d				1		1	
e	1	1			1		
f						1	
g			1				1

	a	b	e	c	g	d	f
a	1	1	1				
b	1	1					
e		1	1				
c				1	1		
g				1	1		
d						1	1
f						1	1

### 20.4.1 Design Structure Matrix and Domain Mapping Matrix

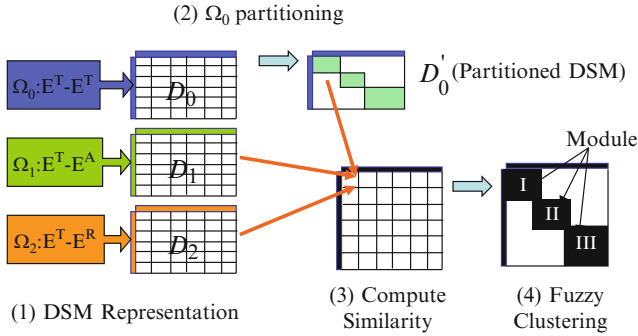
Initially, the DSM is used to represent the activity dependencies, which can be derived from directed graph that shows the interdependency and sequence relationships of the constituent elements (Steward, 1981). Figure 20.5a shows a DSM consisting of seven activities. An element  $m_{ij} = 1$  indicates that activity  $i$  is dependent on activity  $j$ , while  $m_{ij} = 0$  (shown as blank in this figure) indicates that activity  $i$  is not dependent on activity  $j$  (Steward 1981; Eppinger et al. 1994; Chen 2005).

A DSM can be analyzed based on matrix operations, such as partitioning, clustering, and sequencing. In modular design, matrix operations have been developed to maximize the interactions within modules and minimize interactions between modules. By rearranging the components in the above example, three modules are identified: module I (a, b, e), module II (c, g), and module III (d, f), as shown in Fig. 20.5b.

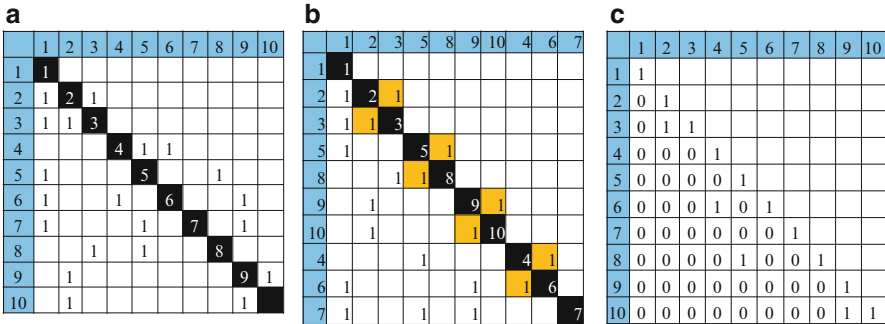
A DMM is a generic form of a DSM, which captures the dependency relationships of two distinctive sets of concepts. Thus, it does not require the row and column of a matrix to be of identical dimension. DMMs can be used to show the other types of relationships, i.e., activity-actor ( $\Omega_1$ ), activities-resources ( $\Omega_2$ ), and actor-resource ( $\Omega_3$ ). For example, to represent  $\Omega_1$ , the first row and the first column of a DMM denote the set of actors and activities, respectively. An element of the DMM denotes whether an activity is carried out by a particular actor.

### 20.4.2 Fuzzy Clustering

Clustering analysis is a process of grouping a set of objects into categories of similar objects (Jiao and Zhang 2005). Fuzzy clustering aims at creating hierarchical decomposition of a set of objects, in which similar objects are successively grouped according to similarity levels. The procedure of fuzzy clustering for module identification is shown in Fig. 20.6.

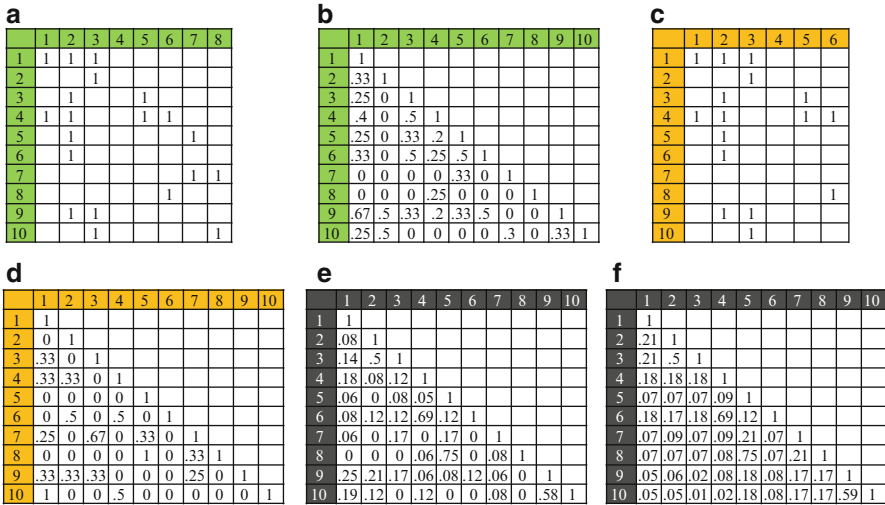


**Fig. 20.6** Fuzzy clustering for module identification



**Fig. 20.7** DSM partitioning and similarity measure. (a) Original DSM ( $D_0$ ). (b) Partitioned DSM ( $D'_0$ ). (c) Similarity based on ( $D'_0$ )

1. Three types of relationships between project elements, namely,  $\Omega_0$ ,  $\Omega_1$ , and  $\Omega_2$ , are represented using binary DSMs/DMMs. Accordingly, the resulting matrices are called  $D_0$ ,  $D_1$ , and  $D_2$ , respectively.
2.  $D_0$  is analyzed using the partitioning algorithm. A partitioning algorithm manipulates the rows and columns of  $D_0$  and groups the objects into a block, such that it does not contain any feedback marks. Thus,  $D_0$  is transformed into a block triangular form, denoted as  $D'_0$ , in which each block represents a module. This can reduce the unnecessary cyclic procedures of the design activities. Many partitioning algorithms have been reported in literature, such as the path searching method (Steward 1981; Gebala and Eppinger 1991), the power of the adjacency matrix method (Warfield 1973), the reachability matrix method (Warfield 1973; Kusiak et al. 1994), and the triangularization algorithm (Kusiak et al. 1994). This research adopts the path search method (Steward 1981) for partitioning  $D_0$ . Figure 20.7a shows a DSM ( $D_0$ ) involving ten activities, which is partitioned into  $D'_0$  with six modules (Fig. 20.7b).



**Fig. 20.8** DSM representation and similarity computation. (a) Activity-player:  $D_1$ . (b) Similarity based on  $D_1$ . (c) Activity-resource:  $D_2$ . (d) Similarity based on  $D_2$ . (e) Combined similarity. (f) Fuzzy equivalence matrix

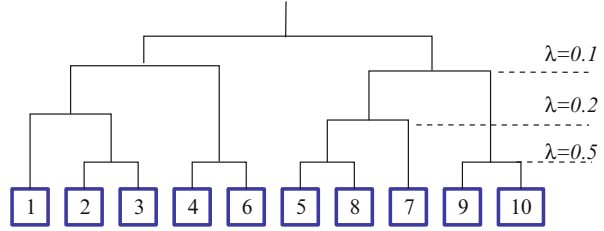
3. The similarity ( $s_{ij}$ ) between each pair of activities (e.g.,  $E_i^T$  and  $E_j^T$ ) is computed from  $D'_0$ ,  $D_1$ , and  $D_2$ :

$$s_{ij} = w_1 s_{ij}^T + w_2 s_{ij}^A + w_3 s_{ij}^R \tag{20.1}$$

where:

- $s_{ij}^T$  is the similarity based on activity sequence and is derived from  $D'_0$ , i.e.,  $s_{ij}^T = D'_0(i, j)$ . The subscripts  $i$  and  $j$  are the index of the activities. Using the same example,  $s_{ij}^T$  is computed as shown in Fig. 20.7c.
- $s_{ij}^A$  is the similarity measure based on actor similarity, which is derived from  $D_1$ , i.e.,  $s_{ij}^A = \frac{AND(D_1(i), D_1(j))}{OR(D_1(i), D_1(j))}$ , where  $D_1(i)$  and  $D_1(j)$  are the row vectors. The dividend  $AND(D_1(i), D_1(j))$  denotes the number of actors that appear in both activities  $i$  and  $j$ , i.e.,  $D_1(i, k) = D_1(j, k) = 1$ . The divisor  $OR(D_1(i), D_1(j))$  denotes the number of actors that appear in either activities, i.e.,  $D_1(i, k) = 1$  or  $D_1(j, k) = 1$ . For example, given the activity-actor matrix  $D_1$  (Fig. 20.8a), the similarity measure is computed as in Fig. 20.8b.
- $s_{ij}^R$  is the similarity measure based on resource similarity, which is derived from  $D_2$ , i.e.,  $s_{ij}^R = \frac{AND(D_2(i), D_2(j))}{OR(D_2(i), D_2(j))}$ . An example is given in Fig. 20.8c, d, where six types of resources are involved.

**Fig. 20.9** Activity clusters resulted from different similarity thresholds



4. A combined similarity measure is computed from the three types of similarity measures. In this chapter, the weights assigned to activity sequence, actors, and resources are  $w_1 = 0.5$ ,  $w_2 = 0.25$ , and  $w_3 = 0.25$ , respectively. Based on the same example, the combined similarity measures are computed using Eq. (20.1), and the results are shown in Fig. 20.8e.

The fuzzy clustering algorithm is used to decompose the activities with respect to the combined similarity matrix (Jiao and Zhang 2005). Given the similarity matrix as shown in Fig. 20.8e, the fuzzy equivalence matrix is computed as shown in Fig. 20.8f. The design activities are clustered according to different similarity thresholds (e.g.,  $\lambda = 0.5, 0.2, 0.1$ ), such that a tree structure is formed as shown in Fig. 20.9. The bottom level MoAs are identified as  $\{1\}$ ,  $\{2\ 3\}$ ,  $\{4\ 6\}$ ,  $\{5\ 8\}$ ,  $\{7\}$ , and  $\{9\ 10\}$ .

The outcome of the module identification based on the fuzzy clustering is used to construct the GVS. Once the GVS is constructed, it can be used for project planning using the genetic algorithms presented in the next section.

## 20.5 Design Process Modeling by Timed Colored Petri Nets

After the design process architecture is built, the next step is to establish a model that integrates various decision variables to generate design process configurations. Petri nets (PNs) with time and color properties are proposed to model the design process and provide decision support to design process planning. The application of PNs in design process modeling requires an unambiguous, one-to-one functional correspondence between PNs and the design process specifications. Colored tokens are adopted to facilitate the generic representation. Moreover, to evaluate the evolutionary characteristics of design processes, the PN model incorporates a time property that denotes the temporal features of the design process.

### 20.5.1 Design Process TCPN Model

A timed colored Petri net (TCPN) model is represented as a seven-tuple:

$$\Psi = \{P, T, C, I, D, m\} \tag{20.2}$$

where

- $P = \{p_i\} (i = 1, 2, \dots, n)$  is a set of places, and a place  $p_i$  is a buffer for holding the actor, resource, or message required/released/created by an activity.
- $T = \{t_i\} (i = 1, 2, \dots, m)$  is a set of transitions, and a transition  $t_i$  denotes an activity.
- $C$  is a finite color set defined on  $P \cup T$ .  $C(P)$  is the color set related to places, and  $C(T)$  is the color set related to transitions. The number of a colored item associated with a place is represented as a token,  $\lambda(C) \in N$ , where  $N$  is a positive integer.
- $I : (P \times T) \rightarrow N$  is the input function that defines directed arcs from places to transitions. If there exists an arc with color  $c \in C$  and weight  $w \in N$  connecting  $p_i$  to  $t_j$ , then  $I(p_i \times t_j)(c) = w$ . The weight denotes the amount of tokens required for firing a transition. The complete set of input places of a transition ( $t_j$ ) is called its preset, denoted as  $\bullet t_j \subseteq P$ .
- $O : (P \times T) \rightarrow N$  is the output function that defines directed arcs from transitions to places. If there exists an arc with color  $c \in C$  and weight  $w \in N$  connecting  $t_j$  to  $p_i$ , then  $O(p_i \times t_j)(c) = w$ . A weight denotes the amount of tokens generated by firing a transition. The complete set of output places of a transition ( $t_j$ ) is called its postset, denoted as  $t_j \bullet \subseteq P$ .
- $D : (T) \rightarrow R^+$  is the firing time function that assigns a nonnegative time delay to every transition.  $R^+$  is a positive real number. The time delay is based on a probability distribution function such that it is non-deterministic.
- The complete set of colored tokens in the places of a PN at a specific time  $\tau$  is called a marking, i.e.,  $m_\tau(c) : (P) \rightarrow N$ .  $m_0(c)$  is called the initial marking.

### 20.5.1.1 TCPN Control

The behavior of the PN is dependent on the high-level features that control the occurrence of events or execution of activities. In turn, the sequence of activities is reflected by the flow of tokens governed by predefined rules.

1. *Enabling rule.* A transition  $t_j$  is enabled by a marking if and only if  $m(p_i)(c) \geq I(p_i, t_j)(c), \forall p_i \in \bullet t_j, \exists c \in C(p_i)$ . Moreover, to enhance the dynamic behavior of the TCPN model, a dependent arc is used, whose weight is controlled by status of particular places.
2. *Firing rule.* An enabled transition can fire, and as a consequence, it removes from each input place the number of tokens equal to the weight of the input arc and deposits in each output place the number of tokens equal to the weight of the output arc. If transition  $t_j$  fires, it is considered as active for a period of time determined by the transition's duration, after which a new marking is generated, i.e.,  $m'(p_i)(c) = m(p_i)(c) + O(p_i, t_j)(c) - I(p_i, t_j)(c), \forall p_i \in P, \forall c \in C(p_i)$ .

The above formulation signifies a rough correspondence between the design process and the PNs. For example, activity ( $E$ ) can be mapped onto the transition set ( $T$ ). The diverse types of tasks ( $E^T$ ), actors ( $E^A$ ), and resources ( $E^R$ ) can be represented by colors. The actual sets of actors and resources that are required for carrying out a task corresponding to the input function  $I : (P \times T)$ . In this scenario, an input place acts as a buffer for holding the colored tokens.

At the same time, the output of a task corresponds to the output function  $O : (P \times T)$ , and accordingly the intermediate items are deposited into the suitable output places upon the completion of an activity. The duration of an activity is governed by the firing time ( $D$ ). Finally, the status of the design process at any time ( $\tau$ ) is recorded by the marking  $m_\tau(c)$ .

### Colored Tokens

Colored tokens are used to differentiate classes of input/output (e.g., CNs, FRs, DPs, and PVs), actors (e.g., customer, executive, engineer), and resources (e.g., equipment, material, facility, budget).

### Time Properties

Time allows for the modeling of temporal behaviors of a design process. A firing time represents the duration of an activity which is estimated based on random variables that follow certain distribution functions. In practice, beta distribution is usually adopted for estimating activity duration based on relevant information extracted from prior projects.

#### 20.5.1.2 Variety Handling Mechanism

By using colored tokens, the structure of PNs is simplified because different activities can be represented by the same structure as long as the functional behaviors of the activity are the same. Figure 20.10 shows an illustrative example of a PN representation of the design process, where mechanisms for dealing with process varieties are handled by color and generic routing.

Two design specifications ( $ds_1$  and  $ds_2$ ) are hosted by place  $p_1$ , representing two product variants. Assume that each design specification is to be fulfilled by distinct components, which can be either selected from existing component catalog (denoted by transition  $t_1$ ) or designed (denoted by transition  $t_2$ ). If transition  $t_2$  (“design new component”) is fired, it is followed by the process of checking the quality of newly designed component (embodied by  $t_3$ ). The compatibility of components is examined in the process denoted by  $t_4$ , resulting in the configuration of product.

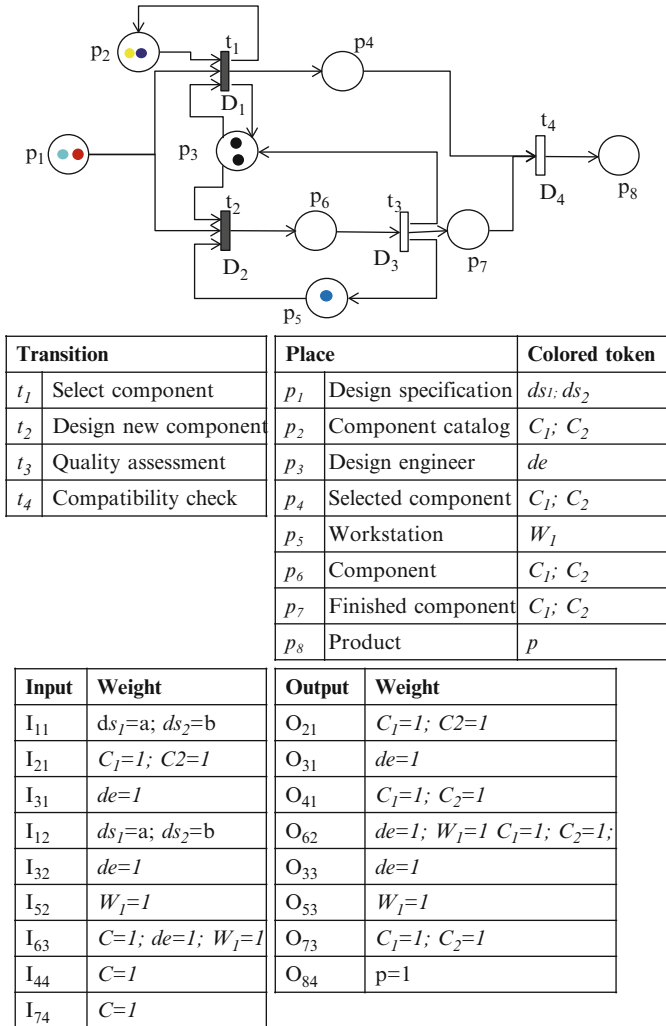
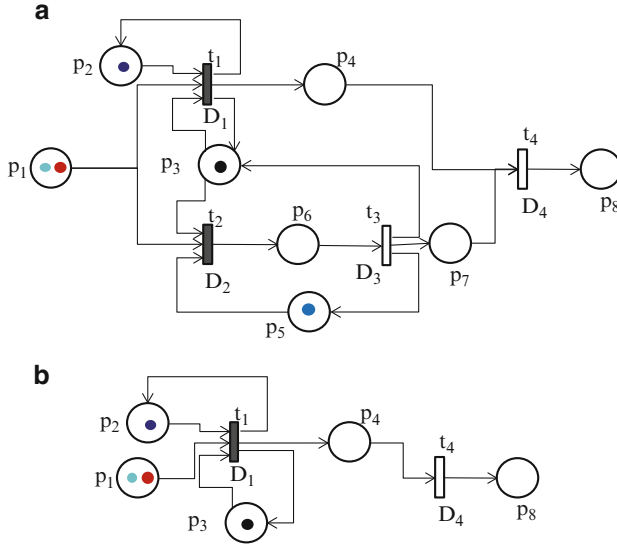


Fig. 20.10 Petri nets for representing design processes

Assume that two products ( $Pd_1$  and  $Pd_2$ ) are to be designed and the design specifications are given as  $\{Pd_1: ds_1 = a_1, ds_2 = b_1\}$  and  $\{Pd_2: ds_1 = a_2, ds_2 = b_2\}$ , respectively. Once the design specifications are loaded to place  $p_1$ , they are compared to the input functions so that a proper routing is selected. For  $Pd_1$ ,  $ds_1: a_1 \leq a$  and  $ds_2: b_1 \geq b$ . Thus, design specification  $ds_1 = a_1$  cannot be fulfilled using the existing component catalog. And therefore,  $ds_1$  is handled by transition  $t_2$  indicating the design of new components.

On the other hand,  $ds_2 = b_1$  falls within the capacity of the existing catalog, and  $ds_2$  is dispatched to transition  $t_1$ , where a designer select a component from the



**Fig. 20.11** Petri nets for handling process varieties. (a) TCPN model of a design process variant  $PC_1$ . (b) TCPN model of a design process variant  $PC_2$

existing catalog. The TCPN model for developing  $Pd_1$  is constructed as process variant  $PC_1$ , as shown in Fig. 20.11a. Similarly, process variant  $PC_2$  is built as Fig. 20.11b, where both design specifications (i.e.,  $ds_1 = a_2$ ,  $ds_2 = b_2$ ) are addressed by the component catalog.

### 20.5.1.3 Performance Indicators

The design process plans are evaluated using two criteria: lead time and cost. For the lead time, this research adopts a lead time penalty function as the objective function. Moreover, considering that cost cannot be precisely estimated at the planning stage, the resource capacity idle rate is used to simulate the cost of product development. The resource is considered busy when it is being occupied or consumed by an activity; otherwise it is idle. For any particular type of resources, the resource capacity idle rate is the percentage of idle time with respect to the overall design process life cycle. In essence, the idle rate of actor and resource is a useful indicator of the cost of product development, e.g., cost can be effectively reduced if the system maintains a low system capacity idle rate (high capacity utilization).

The PN representation of resources using colored tokens provides a convenient way to compute the actor/resource idle rate. An actor/resource is utilized if it is associated with a fired transition. Hence, the amount of resource being utilized equals to the weight of the input arc that connects the preset places and the fired transition. Based on this observation, the utilization rate of actor/resource  $j$  at time can be computed as



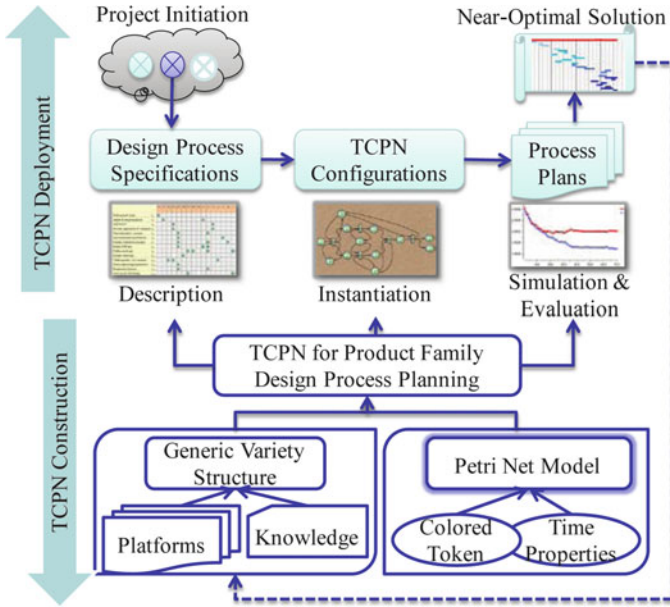


Fig. 20.12 TCPN decision support to design process planning

$$u_j(\tau) = \frac{\sum_{i=1}^m \left( y_i \sum_{k=1}^{n_i} I_j^k(\tau) \right)}{\sum_{i=1}^n \lambda_j^{p_i}(\tau) + \sum_{i=1}^m \left( y_i \sum_{k=1}^{n_i} I_j^k(\tau) \right)} \tag{20.3}$$

where:

- $\lambda_j^{p_i}(\tau)$  is the number of tokens held in a place  $p_i \in P(\forall i = 1, 2, \dots, n)$ .
- $I_j^k(\tau)$  is the weight of the input arc connecting places  $p_k \in \bullet t_i(\forall k = 1, 2, \dots, n_i)$  and transition  $t_i \in T(\forall i = 1, 2, \dots, m)$ .
- $y_i$  is a Boolean value such that  $y_i = \begin{cases} 1, & \text{if } t_i \text{ is activated (fired);} \\ 0, & \text{otherwise.} \end{cases}$

### 20.5.2 Decision Support to Design Process Planning

Based on the TCPN formalism, a decision framework is developed which consists of two major stages, namely, TCPN construction and TCPN deployment, as shown in Fig. 20.12. The former involves the establishment of TCPN model based on knowledge of the product variant design process. The latter aims at applying the TCPN model for decision-making in design process planning.

### 20.5.2.1 TCPN Construction

The procedures shown next provide general guidelines to construct the TCPN:

- *Decompose design process.* This step involves the decomposition of the design process according to the design process architecture, which consists of three elements  $E^T$ ,  $E^A$ , and  $E^R$ . Data maintained in the documentation of an enterprise can be analyzed using such tools as DSM and WBS (Seol et al. 2007; Liu and Horowitz 1989).
- *Construct GVS.* Following the concept of object-oriented modeling, the design process is decomposed into elementary activities. Instances of activities are characterized by specific tasks, actors, and resources.
- *Identify color set.* The color set can be readily extracted from the leaf nodes of GVS. It corresponds to the definition of activity structure in terms of task, actor, and resource.
- *Identify time features.* The expected durations of design activities are estimated from documentations of prior projects.
- *Construct TCPN.* The activities at the leaf node of the GVS are initiated as transitions. Next, the elements (e.g., input/output of tasks, actors, and resources) associated with the leaf node activities are used to define places. Accordingly, the connections between transitions and places are defined. The precedence links and junction relations in the GVS are used to define the dependency relationships between transitions. Places are connected to transitions based on the request or release of actors, resources, and message. Finally, time features are assigned to transitions considering the availability of colored tokens.

### 20.5.2.2 TCPN Deployment

Design process planning starts from the description of project scopes and time/resource constraints. Next, configurations of TCPN for developing the product variants are generated through an instantiation process, which determines the routings of activities and the preliminary assignment of actors and resources, represented as the initial marking of the TCPN. Design process plans are generated and evaluated based on discrete-event simulation. Alternative plans are simulated and the performance of these candidate solutions is evaluated based on the system capacity idle rate ( $U$ ) and the lead time penalty function ( $V$ ), leading to near-optimal solutions.

The TCPN provides a convenient means to generate design process plans through simulation, i.e., through the analysis of behavioral properties of PNs. While the analysis can be carried out using the reachability tree or the invariant methods (Murata 1989), such an analysis does not give intuitive explanations of the results. Hence, a simulation process is more appropriate for studying the evolving status of the system. In this research, a prototype TCPN software package was developed to implement the simulation routines.

## 20.6 Case Study

The proposed approach is implemented for the project planning of car dashboards in an automobile company. The car dashboard is a subassembly with several basic models and a number of customizable features. Product variants can be generated by selecting the appropriate features, which are fulfilled by predefined product components. In order to meet the tight schedules and reduce the product development cost, it is necessary to coordinate the design activities belonging to multiple projects so as to achieve favorable task concurrency and resource reuse. The design process structure is first established using the modularization approach. Next, the TCPN model is adopted to generate design process plans based on a comprehensive design process model and to evaluate their performance based on simulations.

### 20.6.1 Design Process Modularization

Prior projects are analyzed to identify the basic design activities, actors, and resources. A total of 23 basic activities, 10 actors, and 12 resources are identified (Table 20.1). It is recognized that the relationships between these elements are only

**Table 20.1** Activities, actors, and resources involved in car dashboard design

	Content	
Activity	<ol style="list-style-type: none"> <li>1. Basic model selection</li> <li>2. Time analysis</li> <li>3. Diffuser specification</li> <li>4. Ornament selection</li> <li>5. Drawer selection</li> <li>6. Color matching</li> <li>7. Drawer geometry checking</li> <li>8. Hazardous emission check</li> <li>9. Visibility check</li> <li>10. Specification team checking</li> <li>11. Chief design engineer approval</li> <li>12. Select diffuser</li> </ol>	<ol style="list-style-type: none"> <li>13. Select ornament</li> <li>14. Select drawer</li> <li>15. Design diffuser</li> <li>16. Design ornament</li> <li>17. Design drawer</li> <li>18. Component interference checking</li> <li>19. Geometry compatibility checking</li> <li>20. Assembly simulation</li> <li>21. Assembly model building</li> <li>22. Assembly model rendering</li> <li>23. Design specification white paper</li> </ol>
Main actor	<ol style="list-style-type: none"> <li>1. Project manager</li> <li>2. Marketing staff</li> <li>3. Analyst</li> <li>4. Graphic designer</li> <li>5. Customer</li> </ol>	<ol style="list-style-type: none"> <li>6. Interior design engineer</li> <li>7. Quality engineer</li> <li>8. Product planner</li> <li>9. Licensed quality engineer</li> <li>10. Product designer</li> </ol>
Resource	<ol style="list-style-type: none"> <li>1. Desktop</li> <li>2. Laptop</li> <li>3. Printer</li> <li>4. CAD modeler</li> <li>5. Machine</li> <li>6. Wood</li> </ol>	<ol style="list-style-type: none"> <li>7. Plastic</li> <li>8. Textile</li> <li>9. Conference room</li> <li>10. Design studio</li> <li>11. Workshop</li> <li>12. Manual</li> </ol>

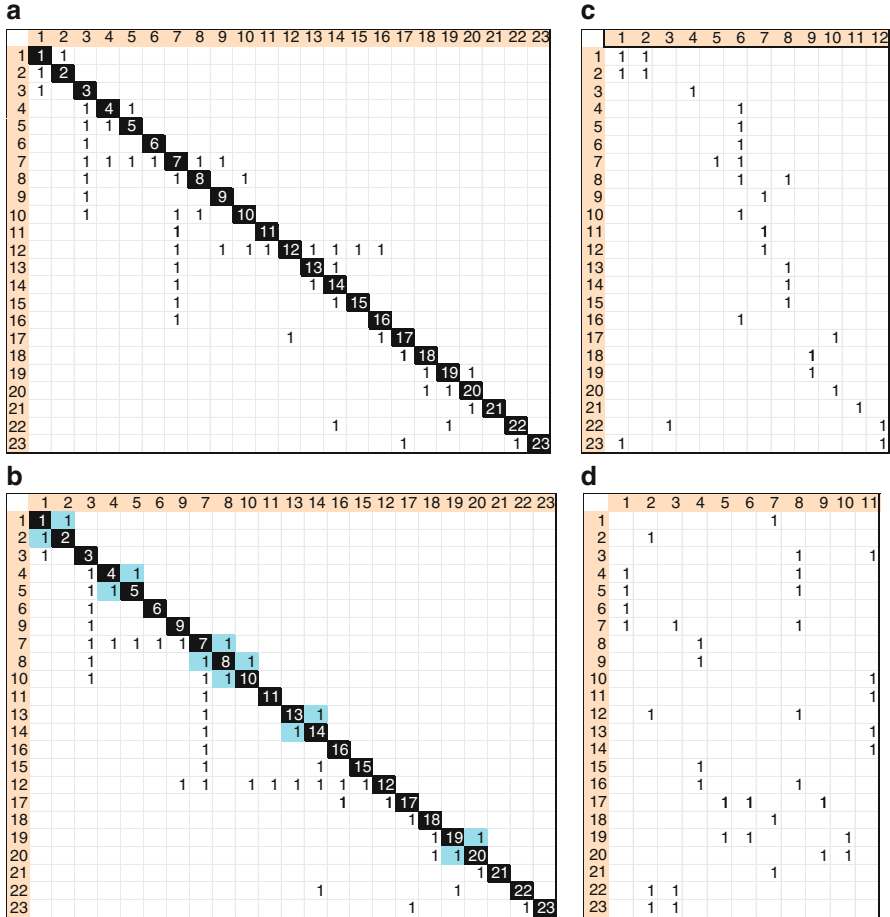


Fig. 20.13 DSM representation of the relationships between project elements. (a)  $D_0$ , (b)  $D'_0$  (partitioned DSM), (c)  $D_1$  and (d)  $D_2$

loosely defined. For example, the actors are roughly grouped into teams corresponding to different stages of product development. Further analysis is required to extract the relationships so as to establish a GVS for the design projects. The dependency relationships  $\Omega_0$  between the activities are represented as a DSM  $D_0$  (Fig. 20.13a). Similarly, the relationship between activities and actors ( $\Omega_1$ ), and activities and resources ( $\Omega_2$ ), are identified as DMMs,  $D_1$  (Fig. 20.13c) and  $D_2$  (Fig. 20.13d). Next,  $D_0$  is processed using the partitioning algorithm, resulting in  $D'_0$  (Fig. 20.13b). From  $D'_0$ ,  $D_1$ , and  $D_2$ , the pairwise similarities between activities are computed as shown in Fig. 20.14. Next, the MoAs are identified using the fuzzy clustering algorithm, resulting in ten MoAs that constitute the GVS (Table 20.2).

The GVS has two layers. In the first layer, a project is roughly decomposed into two stages, namely, requirement analysis and product design. In the second layer,

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
1	1	.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.20	0	0	.20	0	.07	
2	.80	1	0	0	0	0	0	0	0	0	.10	0	0	0	0	0	0	0	0	0	0	.10	.17	
3	0	0	1	.07	.07	0	.05	0	0	.10	.07	.07	.10	.10	0	.05	0	0	0	0	0	0	0	0
4	0	0	.07	1	1	.27	.23	.10	0	.20	0	.07	0	0	0	.25	0	0	0	0	0	0	0	0
5	0	0	.07	1	1	.27	.23	.10	0	.20	0	.07	0	0	0	.25	0	0	0	0	0	0	0	0
6	0	0	0	.27	.23	1	.15	.10	0	.20	0	0	0	0	0	.20	0	0	0	0	0	0	0	0
7	0	0	.05	.23	.23	.15	1	.67	0	.10	0	.05	0	0	0	.14	0	0	0	0	0	.05	.05	
8	0	0	0	.10	.10	.10	.67	1	.10	.10	0	.10	.10	.30	.17	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	.10	1	0	.20	.20	0	0	.10	.05	0	0	0	0	0	0	0	0
10	0	0	.10	.20	.20	.20	.10	.10	0	1	.10	0	.20	.20	0	.20	0	0	0	0	0	0	0	0
11	0	0	.07	0	0	0	0	0	.20	.10	1	.20	.10	.10	0	0	0	0	0	0	0	0	0	0
12	0	.10	.07	.07	.07	0	.05	0	.20	0	.20	1	0	0	0	.05	0	0	0	0	0	0	.07	.07
13	0	0	.10	0	0	0	0	.10	0	.20	.10	0	1	1	.20	0	0	0	0	0	0	0	0	0
14	0	0	.10	0	0	0	0	.10	0	.20	.10	0	1	1	.20	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	.30	.10	0	0	0	.20	.20	1	.07	0	0	0	0	0	0	0	0
16	0	0	.05	.25	.25	.20	.14	.17	.05	.20	0	.05	0	0	.07	1	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	.08	.24	0	0	0	0
18	.20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	.20	0	.20	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.08	.20	1	.65	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.24	0	.65	1	0	0	0	0
21	.20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.20	0	0	1	0	0
22	0	.10	0	0	0	0	.05	0	0	0	0	.07	0	0	0	0	0	0	0	0	0	0	1	.27
23	.07	.17	0	0	0	0	.05	0	0	0	0	.07	0	0	0	0	0	0	0	0	0	0	.27	1

Fig. 20.14 Combined similarity between design activities

Table 20.2 Generic design process variety structure for the dashboard

	Activity	Task	Task		Actor	Resource
			Input	Output		
Requirement analysis	Market demand analysis	$E_1$	$cn_1 \sim cn_3$	$cn_1 \sim cn_3$	$h_1, h_4$	$e_2$
	Initiate design specifications	$E_2$	$cn_1 \sim cn_3$	$ds_1 \sim ds_3$	$h_1, h_3$	$e_1, e_3, f_3$
	Validate compatibility	$E_3$	$ds_1, ds_3$	$ds_1, ds_3$	$h_2$	$e_1, e_3, f_3$
	Check regulation	$E_4$	$ds_3$	$ds_3$	$h_2$	$e_1, f_2$
Product design	Dispatch design specifications	$E_5$	$ds_1 \sim ds_3$	$ds_1 \sim ds_3$	$h_3$	$e_1, e_3$
	Select component	$E_6$	$ds_1 \sim ds_3$	$pc_1 \sim pc_4$	$h_3$	$e_1, m_1 \sim m_4, f_2$
	Design new component	$E_7$	$ds_1 \sim ds_3$	$pc_1 \sim pc_4$	$h_3, h_5$	$e_1, e_3, f_3$
	Quality assessment	$E_8$	$pc_1 \sim pc_4$	$pc_1 \sim pc_4$	$h_2, h_5$	$e_1, e_3, f_2$
	Check compatibility	$E_9$	$pc_1 \sim pc_4$	$pc_1 \sim pc_4$	$h_2$	$e_1, f_2$
	Validate product with customer	$E_{10}$	$pc_1 \sim pc_4$	$pc_5$	$h_4$	$e_1, e_2, f_3$

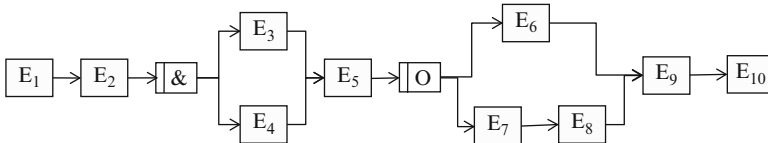
the MoAs are defined in terms of tasks, actors, and resources. For example, market demand analysis ( $E_1$ ) requires two actors, a marketing staff ( $h_1$ ) and customer ( $h_4$ ), and one resource type, a conference room ( $e_2$ ). The detailed information of actors and resources are shown in Tables 20.3 and 20.4, where the last columns show the actor/resource instances.

**Table 20.3** Input/output class (message color set) for dashboard design

Type	Class	Code	Instance
Customer need	Driving guide	$cn_1$	None, smart
	Safety	$cn_2$	Basic, enhanced
	Entertainment	$cn_3$	Basic, good
Functional requirement	Auto parking	$ds_1$	No, yes
	Audio	$ds_2$	Basic, Hi-Fi
	Passive safety	$ds_3$	Basic, advanced
Product component	Distance detector	$pc_1$	No, yes
	Power assist	$pc_2$	No, yes
	Audio system	$pc_3$	Set 1, Set 2
	Air bag	$pc_4$	Basic, side
	Dashboard	$pc_5$	Product

**Table 20.4** Actor and resource for dashboard design

Type	Class	Code	Instance	
Actor	Marketing staff	$h_1$	$h^{*1}_1$	
	Quality engineer	$h_2$	$h^{*1}_2, h^{*2}_2$	
	Product planner	$h_3$	$h^{*1}_3, h^{*2}_3$	
	Customer	$h_4$	$h^{*1}_4, h^{*2}_4$	
	Product designer	$h_5$	$h^{*1}_5$	
Resource	Equipment	Workstation	$e_1$	$e^{*1}_1, e^{*2}_1, e^{*3}_1$
		PDA	$e_2$	$e^{*1}_2, e^{*2}_2$
		Printer	$e_3$	$e^{*1}_3, e^{*2}_3, e^{*3}_3$
	Material	Distance detector	$m_1$	$m^{*1}_1$
		Power assist	$m_2$	$m^{*1}_2$
		Audio system	$m_3$	$m^{*1}_3, m^{*2}_3$
		Air bag	$m_4$	$m^{*1}_4, m^{*2}_4$
	Facility	Conference room	$f_1$	$f^{*1}_1, f^{*2}_1$
		Manual	$f_2$	$f^{*1}_2, f^{*2}_2$
		Design studio	$f_3$	$f^{*1}_3$



**Fig. 20.15** Generic routing of the dashboard design process

Next, a generic routing of the activities is created as shown in Fig. 20.15. The design process starts when a potential customer ( $h_1$ ) consults with the marketing staff ( $h_4$ ), who carries out an analysis of customer’s preferences (activity  $E_1$ ). Based on the customer need information ( $cn_1$ ,  $cn_2$ , and  $cn_3$ ), a product planner ( $h_3$ ) is

responsible for initiating the design specifications ( $E_2$ ), which are technical interpretations ( $ds_1$ ,  $ds_2$ , and  $ds_3$ ) of the CNs. Among the design specifications,  $ds_1$  (auto parking) and  $ds_3$  (passive safety) may conflict with each other, thus necessitating an activity of compatibility validation ( $E_3$ ). At the same time,  $ds_3$  (safety) is critical and must be examined against the regulations in activity  $E_4$ . Once confirmed, all design specifications are dispatched ( $E_5$ ) to planners, who may either select a standard component from the component catalog (activity  $E_6$ ) or design a new component ( $E_7$ ). Whenever  $E_7$  is involved, it is followed by a quality assessment activity ( $E_8$ ), carried out by designers ( $h_5$ ) and quality engineers ( $h_2$ ). After checking the compatibility of components ( $E_9$ ), the product configuration is validated with the customer ( $E_{10}$ ) before it is issued to the production department.

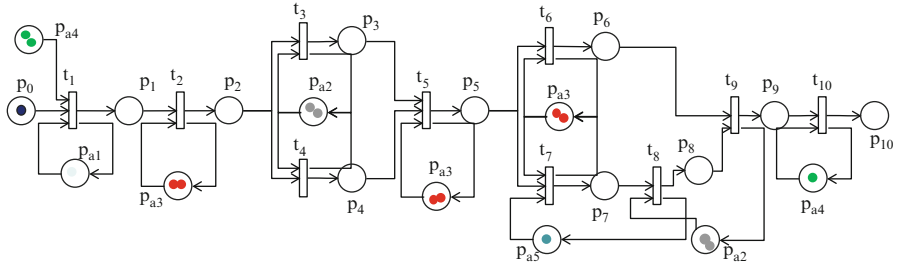
### 20.6.2 TCPN for Design Process Simulation

The GVS is further denoted as the TCPN, as shown in Fig. 20.16. In this model, each transition ( $t_x$ ) strictly corresponds to an activity ( $E_x$ ), and a place is the holder of colored tokens, representing actors, resources, or messages. For purpose of clarity, places for holding resources are hidden in the figure. Places for holding the actor tokens are suffixed by “ $ax$ ,” such that places with identical “ $x$ ” value are multiple copies of the same place. The sets of colors for denoting message, actor, and resource classes are defined in Tables 20.3 and 20.4.

In the TCPN model, place  $p_0$  represents the CNs within the range of product family. Arc  $I_{0-1}(p_0, t_1)$  is a dependent arc, whose weight reflects the level of customer needs as embodied by the content of  $p_{a4}$ . In other words, depending on specific demands of customers (colored token residing in  $p_{a4}$ ), the weight of arc  $I_{0-1}$  is assigned dynamically to reflect the changing demands. Such logic is denoted as  $I_{0-1}: cn_x = p_{a4}.cn_x$ . It provides a mechanism to ensure that only customer orders relevant to the present product family are addressed. Subsequently, the CNs trigger a series of design processes that is described in the GVS. As such, different routings and actor/resource allocation schemes are realized, denoting different plans for product variant design. Without repeating the description of entire design process, the meaning of transitions, places, and input/output arcs is summarized corresponding to the TCPN diagram in Fig. 20.16.

### 20.6.3 Design Process Planning

After the TCPN is constructed, the TCPN deployment stage is activated to embody the product variant generation process. In this case study, two product variants, namely,  $M_1$  and  $M_2$ , are to be designed based on the requests of customers. The expected lead times of these products are 55 and 65 days, respectively.



Transition		Duration		Place		Colored token
		$t_e$ (days)	$\sigma^2$			
$t_1$	Market demand analysis	7	1	$p_0$	Customer need	$cn_{1-3}$
$t_2$	Initiate design specifications	3	0.5	$p_1$	Customer need	$cn_{1-3}$
$t_3$	Validate compatibility	3.3	0.5	$p_2$	Design specification	$ds_{1-3}$
$t_4$	Check regulation	1	0.5	$p_3$	Design specification	$ds_1, ds_3$
$t_5$	Dispatch design specifications	4.5	0.8	$p_4$	Design specification	$ds_3$
$t_6$	Select component	6.5	2	$p_5$	Design specification	$ds_{1-3}$
$t_7$	Design new component	20	5.3	$p_6$	Component	$pc_{1-4}$
$t_8$	Quality assessment	8	2.2	$p_7$	Component	$pc_{1-4}$
$t_9$	Check compatibility	2.7	0.4	$p_8$	Component	$pc_{1-4}$
$t_{10}$	Validate product with customer	6	2.6	$p_9$	Component	$pc_{1-4}$
				$p_{10}$	Product	$p$
				$p_{a1}$	Marketing staff	$h_1$
				$p_{a2}$	Quality engineer	$h_2$
				$p_{a3}$	Product planner	$h_3$
				$p_{a4}$	Customer	$h_4$
				$p_{a5}$	Product designer	$h_3$

- <b>Input arc</b>						
$I_{0,1}: cn_i = p_{a4}, cn_x$	$I_{a1,1}: h_1 = 1$	$I_{a4,1}: h_1 = 1$	$I_{1,2}: cn_i = u_x$	$I_{a3,2}: h_3 = 1$	$I_{2,3}: ds_1 = 0; ds_3 = 0$	
$I_{a2,3}: h_2 = 1$	$I_{2,4}: ds_2 = 0$	$I_{a2,4}: h_2 = 1$	$I_{3,5}: ds_j = v_j; ds_3 = v_3$	$I_{4,5}: ds_2 = v_2$	$I_{a3,5}: h_3 = 1$	
$I_{5,6}: ds_i = v_x$	$I_{a3,6}: h_3 = 1$	$I_{5,7}: ds_x = v_x$	$I_{a3,7}: h_3 = 1$	$I_{a5,7}: h_3 = 1$	$I_{7,8}: ds_x = v_x$	
$I_{a2,8}: h_2 = 1$	$I_{6,9}: pc_x = w_x$	$I_{8,9}: pc_x = w_x$	$I_{9,10}: pc_x = w_x$	$I_{a4,10}: h_4 = 1$		
- <b>Output arc</b>						
$O_{1,1}: cn_x = u_x$	$O_{a1,1}: h_1 = 1$	$O_{2,2}: cn_x = u_x$	$O_{a3,2}: h_3 = 1$	$O_{3,3}: ds_1 = v_1; ds_3 = v_3$	$O_{a2,3}: h_2 = 1$	
$O_{4,4}: ds_2 = v_2$	$O_{a2,4}: h_2 = 1$	$O_{5,5}: ds_x = v_x$	$O_{a3,5}: h_3 = 1$	$O_{6,6}: pc_x = w_x$	$O_{a3,6}: h_3 = 1$	
$O_{7,7}: pc_x = w_x$	$O_{a3,7}: h_3 = 1$	$O_{a5,8}: h_3 = 1$	$O_{8,8}: pc_x = w_x$	$O_{9,9}: pc_x = w_x$	$O_{a2,9}: h_2 = 1$	
$O_{10,10}: pc_5 = w_5$	$O_{a4,10}: h_4 = 1$					
(x=1,2,3; $u_x, v_x, w_x$ are cardinal numbers)						

Fig. 20.16 Petri net for modeling the design process of the dashboard product family

The process of generating product variants starts with the initialization of the TCPN model. It involves the creation of a routing plan and a preliminary assignment of actors and resources to the respective places. Two alternative routings are generated (Fig. 20.17), which differ in terms of the inclusion or exclusion of design activities embodied by transitions  $t_7$  (i.e., designing new components) and  $t_8$  (i.e., quality assessment). The first routing does not involve the process of designing new components, such that all design specifications can be fulfilled by the legacy component catalog. However, such a process appears in the second



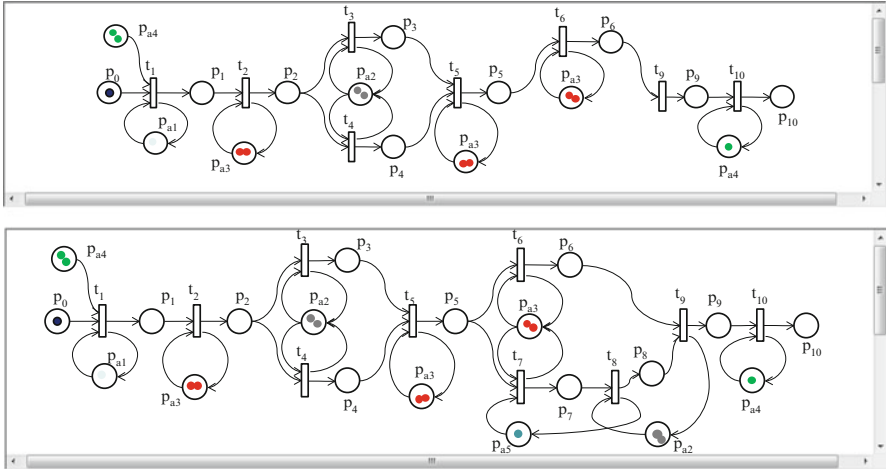


Fig. 20.17 Routings of two process variants

routing, in hope of improving system resource utilization and product quality. Both configurations are potentially capable of fulfilling the task requirements, whereas they may lead to different performance of design process plans.

The actual design process plans are generated through the simulation process of the TCPN. A prototype TCPN software package was developed to implement the routines. Though the simulation, the status of the system is examined and updated as the time stamp increases. The simulation terminates when the entire planning process is completed, which is indicated by the creation of product configurations in place  $p_{10}$ .

### 20.6.4 Results and Analysis

Based on the above simulation process, two design process plans,  $PL_1$  and  $PL_2$ , are generated from the two TCPN configurations. The durations of the activities in both plans are illustrated as the Gantt chart in Fig. 20.18. The allocation of actors and resources is summarized in Table 20.5, where instances of actors and resources are specified. Based on the TCPN simulation results, the performance of the two candidate design process plans is summarized in Table 20.6, showing the average system resource idle rates and the lead time penalty functions.

The lead times for developing two product models are 46 and 65 days in  $PL_1$ , and 51 and 59 days in  $PL_2$ . Assuming that the two products have the same priority, the lead time penalty functions can be computed according to Eq. (20.3), i.e.,  $V_1 = 0.354$  and  $V_2 = 0.355$ . From these results, it is observed that the difference between them is negligible, indicating that both plans confirmed approximately to the expected time line. On the other hand,  $PL_2$  outperforms  $PL_1$  in terms of overall system capacity utilization. In particular, the system capacity idle rates for two

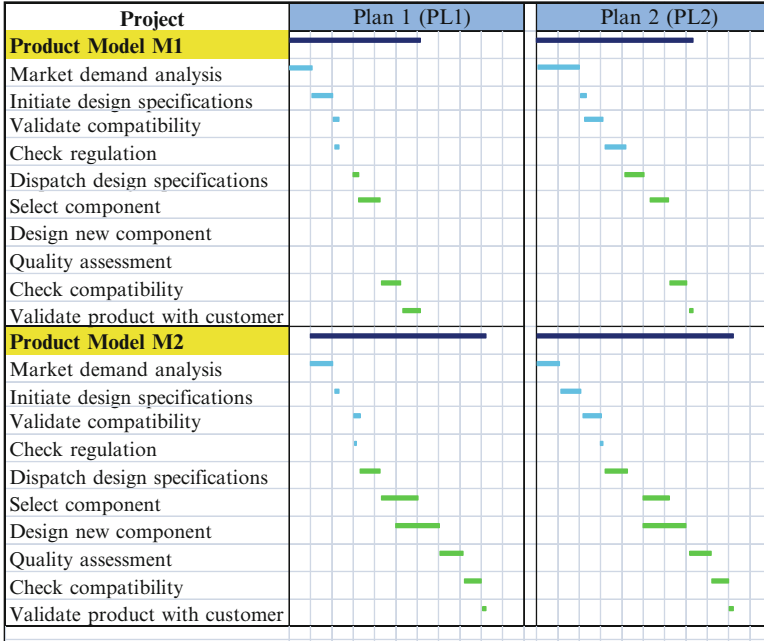


Fig. 20.18 Gantt chart of the design process plans

plans are 0.51 and 0.38, respectively. As shown earlier, the major structural difference between the two TCPN configurations lies in the inclusion/exclusion of the new component design and quality assessment process ( $E_7$  and  $E_8$ ). When both product variants are designed based on existing component catalog (as is in  $PL_1$ ), the design process will compete for the same set of actors ( $h_3$ ) and resources ( $e_1, m_3, m_4, f_2$ ), indicating potential process conflicts. Meanwhile, the set of actors ( $h_2$  and  $h_5$ ) and resource ( $e_3$ ), which are used in processes  $E_7$  and  $E_8$ , stay idle. By including the process of  $E_7$  and  $E_8$ , these idle actors and resources are utilized, such that the workload is allocated in a more balanced way. Such a difference indicates that  $PL_2$  achieves lower cost of product development. Considering that the lead time penalty functions are identical in the two plans,  $PL_2$  achieves a higher cohort performance for developing the product variants.

## 20.7 Discussions

Design process planning presents a new perspective of product variant design. Instead of focusing on product fulfillment through platform-based techniques, this research streamlines the design processes so that they are better controlled in terms of time and cost. Such an effort is appropriate due to the activity concurrency and

**Table 20.5** Actor and resource allocation for two design process plans

Activity	Process plan $PL_1$		Process plan $PL_2$		
	Actor	Resource	Actor	Resource	
Product variant (P1)	$E_1$	$h^{*1}_1, h^{*1}_4$	$e^{*1}_2$	$h^{*1}_1, h^{*2}_4$	$e^{*1}_2, e^{*2}_2$
	$E_2$	$h^{*1}_1, h^{*2}_3$	$e^{*3}_1, e^{*3}_3, f^{*1}_3$	$h^{*1}_1, h^{*2}_3$	$e^{*2}_1, e^{*3}_3, f^{*1}_3$
	$E_3$	$h^{*1}_2$	$e^{*1}_1, e^{*2}_3, f^{*1}_3$	$h^{*1}_2$	$e^{*1}_1, e^{*2}_3, f^{*1}_3, f^{*2}_3$
	$E_4$	$h^{*1}_2$	$e^{*3}_1, f^{*1}_2$	$h^{*2}_2$	$e^{*1}_1, e^{*2}_1, f^{*1}_2$
	$E_5$	$h^{*1}_2$	$e^{*3}_1, e^{*3}_3$	$h^{*1}_2$	$e^{*3}_1, e^{*3}_3$
	$E_6$	$h^{*1}_3, h^{*2}_3$	$e^{*2}_1, m^{*1}_3, m^{*1}_4, f^{*2}_2$	$h^{*1}_3$	$e^{*2}_1, m^{*1}_3, m^{*1}_4, f^{*2}_2$
	$E_7$	—	—	$h^{*1}_3, h^{*2}_5$	$e^{*1}_1, e^{*3}_3, f^{*1}_2$
	$E_8$	—	—	$h^{*1}_2, h^{*1}_5$	$e^{*1}_1, e^{*3}_3, f^{*1}_2$
	$E_9$	$h^{*1}_2$	$e^{*1}_1, f^{*1}_2$	$h^{*1}_2$	$f^{*1}_1, f^{*1}_2$
	$E_{10}$	$h^{*1}_4$	$e^{*2}_1, e^{*2}_2, f^{*1}_3$	$h^{*2}_4$	$e^{*2}_1, e^{*2}_2, f^{*1}_3$
Product variant (P2)	$E_1$	$h^{*1}_1, h^{*2}_4$	$e^{*1}_2, e^{*2}_2$	$h^{*1}_1, h^{*2}_4$	$e^{*1}_2, e^{*2}_2$
	$E_2$	$h^{*1}_1, h^{*1}_3$	$e^{*3}_1, e^{*2}_3, f^{*1}_3$	$h^{*1}_1, h^{*1}_3$	$e^{*2}_1, e^{*3}_3, f^{*1}_3$
	$E_3$	$h^{*2}_2$	$e^{*3}_1, e^{*3}_3, f^{*1}_3$	$h^{*2}_2$	$e^{*1}_1, e^{*2}_3, f^{*1}_3, f^{*2}_3$
	$E_4$	$h^{*2}_2$	$e^{*3}_1, f^{*1}_2$	$h^{*2}_2$	$e^{*1}_1, e^{*2}_1, f^{*2}_2$
	$E_5$	$h^{*2}_2$	$e^{*3}_1, e^{*3}_3$	$h^{*1}_2, h^{*2}_2$	$e^{*3}_1, e^{*3}_3$
	$E_6$	$h^{*1}_3, h^{*2}_3$	$e^{*2}_1, m^{*1}_1, m^{*1}_2, m^{*1}_3, m^{*2}_4, f^{*2}_2$	$h^{*2}_3$	$e^{*2}_1, m^{*1}_1, m^{*1}_2, m^{*1}_3, m^{*1}_4, f^{*2}_2$
	$E_7$	—	—	$h^{*2}_3, h^{*1}_5$	$e^{*1}_1, e^{*3}_3, f^{*1}_2$
	$E_8$	—	—	$h^{*1}_2, h^{*1}_5$	$e^{*1}_1, e^{*2}_3, f^{*2}_2$
	$E_9$	$h^{*2}_2$	$e^{*1}_1, f^{*2}_2$	$h^{*1}_2$	$f^{*1}_1, f^{*1}_2$
	$E_{10}$	$h^{*1}_4$	$e^{*2}_1, e^{*1}_2, f^{*1}_3$	$h^{*2}_4$	$e^{*2}_1, e^{*2}_2, f^{*1}_3$

**Table 20.6** Performance of two design process plans

Plan	Resource utilization		Lead time				Penalty function
	Utilization rate ( $U$ )	Idle rate ( $1-U$ )	Expected ( $\tau^e$ )		Actual ( $\tau^c$ )		
$PL_1$	0.49	0.51	$P1$	$P2$	$P1$	$P2$	0.354
$PL_2$	0.62	0.38			51	59	0.355

reusability inherent in the process of product variant development. Based on an analysis of the design process architecture, the essential elements of design processes are identified as tasks, actors, and resources. It establishes a multiple viewpoint of the design process information, which is an improvement to traditional methods that adopt a single viewpoint.

To tackle the large number of design process varieties, a GVS is constructed to denote the design process variants. In this respect, the multiple process variants can be instantiated from the same model, and the design activities can be better coordinated within the common framework. Furthermore, the modular design process method addresses module identification and project reconfiguration. In particular, a DSM-based method provides analytical rigor to construct the modular project structure. A fuzzy clustering algorithm is capable of constructing a hierarchical activity decomposition tree, accommodating various sources and multiple levels of modularity.

The TCPN model provides decision support in design process planning, which is characterized by a generic structure based on colors and a task-driven planning based on time. The TCPN is advantageous to traditional PN-based design process planning methods because it saves the effort of generating distinct Petri nets for each and every design process variant.

The method proposed in this research is a useful step toward modeling the design process of product families. While the method is directed toward the product variant design process, i.e., focusing on the deployment stage of product family design, it can be readily extended to the development of product architectures, i.e., the construction stage of product family design. This is because the basic elements used in the design process model, e.g., activities, tasks, actors, resources, and generic routings, are independent of the content of the actual design process. In other words, the process of constructing product architectures also involves a series of activities, carried out/supported by shared, reusable actors and resources. In this respect, this research establishes the foundation of managing design process for product families, which exploits the generic nature of the design process elements and the common sharing logic.

## 20.8 Summary

Product variant design facilitates an increasing number of product varieties, which causes a proliferation of design processes throughout the product development life cycle. This research proposes a modularization approach to support design project modeling and planning for product families. Modularizing the design projects facilitates the organization of design activities so as to simplify the decision process in project planning. A project model that consists of three major elements (activity, actor, and resource) is proposed to represent the project information, which enhances information modeling for project management. Based on the project model, a GVS provides a common modular framework that considers task sequences, actor, and resource relationships. The GVS can accommodate the requirements of product family design projects by modeling the project varieties in a consistent and concise manner. Furthermore, the modular approach addresses module identification and project reconfiguration. In particular, a DSM- and DMM-based method provides analytical rigor to construct the modular design process structure. A fuzzy clustering algorithm is capable of constructing a hierarchical activity decomposition tree, accommodating various sources and multiple levels of modularity.

A timed colored Petri net is proposed for modeling and analyzing the product variant design process. As an extension of the platform-based product development, the modeling framework exploits the commonalities inherent in product variant design process. Although such commonality is not necessarily equivalent to product commonality, it is worthwhile to investigate whether such commonality can be used for planning intelligence. Moreover, a TCPN decision framework is developed to integrate the reasoning logic of design process planning. The diverse types of design process

information can be captured using a generic color set. Finally, the cohort performance of the design process plans is evaluated using discrete-event simulation. The application of TCPN in car dashboard family design demonstrates that the method is conducive to managing product design from a systematic level and coordinating activities belonging to multiple projects to achieve favorable resource sharing and reuse.

## References

- Alizon F, Khadke K, Thevenot HJ, Gershenson JK, Marion TJ, Shooter SB, Simpson TW (2007) Frameworks for product family design and development. *Conc Eng Res Appl* 15:187–199
- Balakrishnan A, Brown S (1996) Process planning for aluminum tubes: an engineering-operations perspective. *Oper Res* 44:7–20
- Browning T (2002) Process integration using the design structure matrix. *Syst Eng* 5:180–193
- Chen SJ (2005) An integrated methodological framework for project task coordination and team organization in concurrent engineering. *Conc Eng Res Appl* 13:185–197
- Chen S-J, Li L (2003) Decomposition of interdependent task group for concurrent engineering. *Comput Ind Eng* 44:435–459
- Eppinger SD, Whitney DE, Smith RP, Gebala DA (1994) A model-based method for organizing tasks in product development. *Res Eng Des* 6:1–13
- Fernandez C (1998) Integration analysis of product architecture to support effective team co-location, Master's thesis, Massachusetts Institute of Technology, Cambridge, MA
- Fixson S (2007) Modularity and commonality research: past developments and future opportunities. *Conc Eng Res Appl* 15:85–111
- Gebala D, Eppinger S (1991) Methods for analyzing design procedures. In: *Proceeding of the ASME 3rd international conference on design theory and methodology*, vol 31, pp 227–233
- Gonzalez-Zugasti JP, Otto KN, Baker JD (2001) Assessing value in platformed product family design. *Res Eng Des* 13:30–41
- Hegge HMH, Wortmann JC (1991) Generic bill-of-material: a new product model. *Int J Prod Econ* 23:117–128
- Jiang P, Shao X, Qiu H, Li P (2008) Interoperability of cross-organizational workflows based on process-view for collaborative product development. *Conc Eng Res Appl* 16:73–87
- Jiao J, Zhang Y (2005) Product portfolio identification based on association rule mining. *Comput Aided Des* 37:149–172
- Jiao J, Tseng MM, Ma Q, Zou Y (2000) Generic bill of materials and operations for high-variety production management. *Conc Eng Res Appl* 8:297–322
- Kao HP, Wang W, Dong J, Ku KC (2006) An event-driven approach with makespan/cost tradeoff analysis for project portfolio scheduling. *Comput Ind* 57:379–397
- Kumar AVK, Ganesh LS (1998) Use of petri nets for resource allocation. *IEEE Trans Eng Manage* 45:49–56
- Kusiak A, Wang J (1993a) Decomposition of the design process. *J Mech Des* 115:687–695
- Kusiak A, Wang J (1993b) Efficient organizing of design activities. *Int J Prod Res* 31:753–769
- Kusiak A, Larson TN, Wang J (1994) Reengineering of design and manufacturing processes. *Comput Ind Eng* 26:521–536
- Leger JB, Morel G (2001) Integration of maintenance in the enterprise: towards an enterprise modeling-based framework compliant with proactive maintenance strategy. *Prod Plan Control* 12:176–187
- Liu LC, Horowitz E (1989) A formal model for software project management. *IEEE Trans Softw Eng* 15:1280–1293

- Mayer RJ, Menzel CP, Painter MK, deWitte PS, Blinn T, Perakath B (1995) Information integration for concurrent engineering (IICE): IDEF3 process description capture method report. Knowledge Based Systems, College Station, TX
- Meier C, Yassine AA, Browning TR (2007) Design process sequencing with competent genetic algorithms. *ASME J Mech Des* 129:566–585
- Meredith JR, Mantel SJ (2003) *Project management: A managerial approach*, 5th ed., John Wiley & Sons, New York
- Mittal S, Frayman F (1989) Towards a generic model of configuration tasks. In: *Proceedings of the international joint conference on artificial intelligence*, Detroit, MI, pp 1395–1401
- Murata T (1989) Petri nets: properties, analysis and applications. *Proc IEEE* 77:541–580
- Park H, Cutkosky MR (1999) Framework for modeling dependencies in collaborative engineering processes. *Res Eng Des* 11:84–102
- Pimpler TU, Eppinger SD (1994) Integration analysis of product decompositions. In: *ASME 6th international conference on design theory and methodology*, Minneapolis, MN, Sept
- Raposo AB, Magalhaes LP, Ricarte ILM (2000) Petri nets based coordination mechanisms for multi-workflow environments. *Comput Syst Sci Eng* 15:315–326
- Ross D (1977) Structured analysis (SA): a language for communicating ideas. *IEEE Trans Software Eng* 3:16–31
- Seol H, Kim C, Lee C, Park Y (2007) Design process modularization: concept and algorithm. *Conc Eng Res Appl* 15:175–186
- Sharman D, Yassine A, Carlile P (2002) Characterizing modular architectures. In: *Proceedings of ASME design engineering technical conferences, DETC2002/DTM-34024*, Montreal, Canada
- Steward DV (1981) The design structure system: a method for managing the design of complex system. *IEEE Trans Eng Manage* 28:71–74
- Suh N (1990) *Axiomatic design: advances and applications*. Oxford University Press, New York, NY
- Upton DM, McAfee AP (2000) A path-based approach to information technology in manufacturing. *Int J Technol Manage* 20:354–372
- van der Aalst WMP, van Hee KM (1996) Business process redesign: a petri-net-based approach. *Comput Ind* 29:15–26
- van Veen EA (1992) *Modeling product structures by generic bills-of-materials*. Elsevier, New York, NY
- Warfield JN (1973) Binary matrices in system modeling. *IEEE Trans Syst Man Cybern* 3:441–449
- Watanabe C, Ane BK (2004) Constructing a virtuous cycle of manufacturing agility: concurrent roles of modularity in improving agility and reducing lead time. *Technovation* 24:573–583
- Wiest J, Levy F (1977) *A management guide to PERT/CPM*. Prentice-Hall, Englewood Cliffs, NJ
- Xu Q, Jiao J (2010) Design project modularization for product families. *ASME J Mech Des* 131:061009
- Yassine A, Braha D (2003) Complex concurrent engineering and the design structure matrix method. *Conc Eng Res Appl* 11:165–176
- Yu T, Yassine A, Goldberg D (2003) Genetic algorithm for developing modular product architectures. In: *Proceedings ASME 15th international conference on design theory and methodology*, Chicago, IL, 2003
- Zurawski R, Zhou MC (1994) Petri nets and industrial applications: a tutorial. *IEEE Trans Ind Electron* 41:567–583

# Chapter 21

## Design Principles for Reusable Software Product Platforms

Carlos O. Morales

**Abstract** This chapter is an introduction to software design based on general principles and engineering techniques, applicable to the design of software platforms that are to be used as a foundation for the construction of product families. The second section focuses on the fundamental characteristics of Design for Change, which describes the qualities and structure that the software system must exhibit as a product platform in order to be capable of evolving and adapting to the changing needs of a range of related products within a family, or within an industry domain over a long period of time. The third section introduces a number of software engineering techniques that ensure that the software system to be developed will have a structure suitable for supporting the general platform qualities presented in the first section. The main purpose of this chapter is to provide the background information on software engineering terms and concepts that are needed to understand Chap. 26.

### 21.1 Introduction

Software engineering is the field of computer science that deals with the design and production of large software systems involving a team of engineers. Computer programming is an implementation activity that involves one isolated person only. Software engineering is essentially a team activity.

In comparison to classic engineering disciplines like electrical, mechanical, or civil engineering, software engineering has not yet reached the same level of maturity, although it is now getting close, after the advent of model-based software development, and design, analysis, simulation, and verification tools are still being

---

C.O. Morales (✉)  
Animas Corporation, Johnson & Johnson Medical Devices,  
200 Lawrence Drive, West Chester, PA 19380, USA  
e-mail: [carlos.o.morales@ieee.org](mailto:carlos.o.morales@ieee.org)

evolved. As an emerging engineering discipline, the development of software must be based on sound principles. The concepts presented in this chapter are the foundation upon which software product platforms are built.

## 21.2 Software Qualities

Just like any engineering activity, software engineering is concerned with building a product. In our case, the product is a software system. In some ways, software products are similar to other engineering products, but software has some striking differences, especially because it is intangible, extremely malleable, and its production is necessarily human intensive.

Because of its malleability, some people think that changing software is easy, but this misconception generally underestimates the complexity of software. To clearly see this point, we just have to recall how other engineering products are changed, for example, bridges, airplanes, and electronic circuits. All these other engineering products are not changed without first making a formal design change and analyzing the corresponding ripple effects of their impact before the implementation. After this, management approval is sought, and only then, the change is actually implemented. However, the change process is not complete until the correctness of the change has been verified. A change in software must be viewed as a change in the design, rather than just rewriting technical words in the code.

Now, let us examine the qualities that are pertinent to software products and software production processes. These qualities are the general goals in the practice of software engineering and affect all the system stakeholders.

### 21.2.1 *Product and Process Qualities*

We use a process to produce the software product. Sound process qualities, therefore, yield good product qualities. A software product is more than an executable code and a user's manual. It includes all the artifacts produced. These include functional requirements, design, source code, and test data.

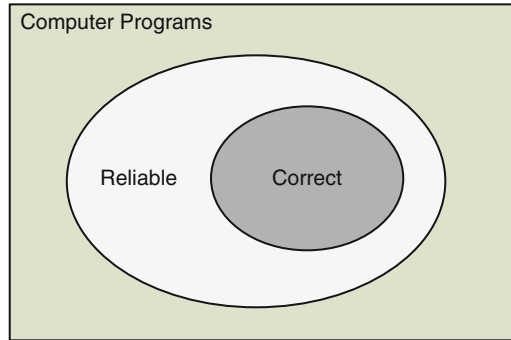
Some of the most relevant software qualities include correctness, reliability, robustness, maintainability, repairability, evolvability, reusability, portability, and interoperability (Ghezzi et al. 1991).

### 21.2.2 *Correctness*

A program that is *correct* behaves according to its functional specification. Specifications, however, could be incorrect, and yet, computer programs that implement that specification faithfully are still correct. This definition of



**Fig. 21.1** All reliable programs include the set of correct programs, but not vice versa



correctness assumes that it is possible to determine in an unambiguous manner whether or not a program meets the specifications. This implies that a functional requirement specification document exists, and correctness establishes the equivalence between the software and its specification.

### 21.2.3 Reliability

A software system that is reliable is one that users can depend on. Reliability as an engineering term is defined in the research literature in terms of statistical behavior—i.e., the probability that the software being evaluated will operate according to the user’s expectations over a specified time interval. The software quality of correctness is absolute, and the most insignificant deviation from the requirements specification renders the system incorrect. In contrast, reliability of a computer program is relative, and a programming error affecting its reliability may be triaged according to its severity. If the consequence of a software error is not serious, the incorrect software may still be reliable.

Engineering products are *expected* to be reliable. Unfortunately, many software products have not achieved this status yet. Users of software take it for granted that “Release 1” of any product is “buggy.” One of the symptoms of the immaturity of software engineering is that a significant number of new products are still released along with a list of “known bugs.” Other engineering products (electronics, cars, airplanes, etc.) are not released if they have “bugs” because they are deemed unacceptable in these circumstances.

Figure 21.1 illustrates the relationship between correctness and reliability. It is assumed that the functional requirements indeed capture the desirable properties of the application, and no undesirable properties are erroneously introduced. All the software can do is to meet the specified requirements of the model. The implementation itself cannot assure the accuracy of the model.

### 21.2.4 Robustness

A computer program is said to be robust if it is able to handle error conditions in a safe manner, even under circumstances that were not anticipated and captured as part of the requirements specification, for example, upon a hardware failure. A program implementation that assumes it will always receive inputs exactly as expected all the time and then crashes at the first error is not robust, even though it might be correct according to its stated requirements specification. Robustness may be better understood with an analogy: two bridges, for example, may be both correct according to their own specification, but if after a storm one of them collapses, then that one is not robust. Correctness, reliability, and robustness also apply to the software production process.

### 21.2.5 Maintainability

The term *maintainability* commonly refers to the quality of software that makes it easy to be modified in order to fix software defects, especially after it has been deployed to the field. Software maintenance costs are very significant in business, since they typically exceed 60 % of the total cost of a software product.

Software maintenance is divided into three categories: corrective, adaptive, and perfective. Corrective maintenance is concerned with the removal of residual errors (bug fixes) that have usually escaped verification of the product. These repairs typically account for about 20 % of the total cost of software maintenance. Adaptive and perfective maintenance are the real sources of evolutionary change necessary in software, and this need motivates the introduction of *evolvability* as a fundamental software quality and *anticipation of change* as a fundamental principle.

Adaptive maintenance involves adjusting the application to changes in technology or its operating environment, for example. Perfective maintenance involves making significant changes to the software in order to improve its qualities, like adding new functions or replacing algorithms that improve its performance. We, therefore, view maintainability as two separate qualities: *repairability* and *evolvability*.

#### 21.2.5.1 Repairability

Repairability is very important in most engineering products like automobiles or computers. Parts that are most likely to fail must be the most accessible. Repairs are typically performed by replacing components that have worn out; like electronic boards, timing belts, or plastic gears.

Software is different since software components do not deteriorate. A software product that consists of well-designed modules is easier to repair than a monolithic entanglement of code. Good modularization promotes repairability by confining errors to a few modules in a larger structure. Good modularization techniques include *information hiding* and *abstract data types*.

### 21.2.5.2 Evolvability

Similar to other products, software products are modified during their lifetime to include new or improved functions. The fact that software is so malleable makes changes very easy to apply to existing code, oftentimes without much consideration as to the ripple effect a seemingly minor change will have across the system. In a rigorous and formal development process, modifications begin at the design level and then proceed to the implementation of the product, but in the case of software, unfortunately, informal code development lacking discipline is still widespread, except in regulated software industries developing safety-critical systems such as medical, avionics, defense, or automobiles. In many of those informal software development teams, it is too frequent that once the change is accomplished, the modification is not even documented a posteriori. When product specifications and design documents are not updated to reflect changes made to the implementation, future changes become more and more difficult.

Formal and rigorous software development yields, in contrast, successful software products with very long lives in the field. Examples of this kind of systems include operating systems, Word Processors, Spreadsheets, and CAD Software. Another incentive for evolvability is the leverage of significant investments in the development of a software product.

### 21.2.6 Reusability

Reusability is directly related to evolvability. When we evolve a product, we modify an existing product to create a new version. If the design is adequate, a completely new product may be created by reusing parts of other products previously developed.

Reusability is typically more applicable to software components than to whole products, but here we aim at software reuse at a larger scale; in the form of software product platforms, as described in Chap. 26. The creation of reusable software components is a good economical decision for many reasons, including that software components allow engineers to design, build, and deploy new products in a shorter time. This approach implies the design and development of self-contained components, featuring standardized interfaces that are to be used as subassemblies in larger systems.

Applied to software engineering, this practice enables developers to build new applications using components that are available off the shelf, have been fully debugged, tested, and are field proven. Ideally, components can be reused in new applications with little or no modification at all. These software components become assets, which are a source of cost savings in the form of improved efficiency, shorter development time, shorter test and debug time, enhanced product quality, a minimized number of software bugs, and delivery of better, more

complete software products to customers. In addition to lower development cost, multiple uses of software assets improve the overall Return On Investment (ROI) of the software development group.

Reusability is very difficult to achieve after a software component is finished and delivered to the field. Instead, reusability must be designed into the new component from the outset. Design for reuse extends to the software development process too, where experience from previous projects must be leveraged. Mature engineering disciplines show good examples of reusable components, such as the optics, electronics, aeronautics, and automobile industries.

### ***21.2.7 Portability***

Software is said to be portable if it has the ability to run in different environments. Environments for software are different processors or operating systems. This quality of software is now being achieved most notably with the Java™ programming language and its virtual machine, which runs on a variety of operating systems, including corporate servers, desktop computers, and embedded platforms. Microsoft's .NET technology promises similar portability to any system supporting the .NET framework. Portability can also be achieved at the source code level with standardized programming languages such as C or C++, although it is sometimes not as straightforward as it could be expected.

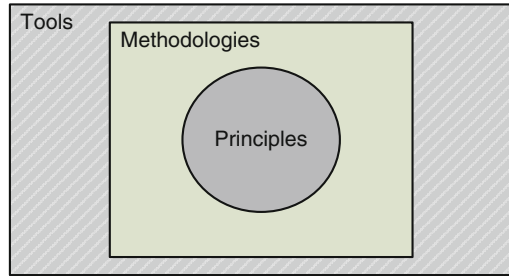
### ***21.2.8 Interoperability***

Interoperability is the ability of a software system to work with other external systems (software or otherwise) in a consistent, efficient, and effective manner. Interoperability is a well-known quality of other mature products, such as audio or video systems, for example, where products from different manufacturers work together seamlessly through standardized interfaces (cables, plugs, data formats, etc.). This ability allows products to accommodate newer technologies as they become available. Interoperability can be achieved in software through the standardization of software component interfaces.

## **21.3 Software Engineering Principles**

The following general software engineering principles are the foundation for successful software development. In order to apply these abstract and general principles, the software engineer needs appropriate methods and techniques, or

**Fig. 21.2** Methodologies and tools



methodology (Fig. 21.2). Special tools are required to support the application of a specific methodology.

Some of the most relevant software engineering principles are *rigor and formality, separation of concerns, modularity, abstraction, design for change, generality, and incrementality*, as originally defined by (Ghezzi et al. 1991).

### 21.3.1 Rigor and Formality

As in many creative activities, there is a tendency in software development to be neither precise nor accurate when set to the task of designing or writing software. Rigor, in the form of discipline, is a necessary ingredient to manage, direct, and improve the engineer's creativity. The highest degree of rigor is what is known as "formal." Formality requires the software process to be driven and evaluated by mathematical laws. Formality implies rigor, but not vice versa.

It is not necessary to be always formal during software design, but it is important to know how and when. An example of the various degrees of formality may be the construction of different types of bridges: the design and construction of a short bridge may be based on rules of thumb. A temporary, larger bridge may use a mathematical model. An exceptionally long bridge in a seismic area would necessarily make use of sophisticated mathematical models and simulation.

In software engineering, requirements specifications must be rigorous, sometimes formal. Software design sometimes requires formalisms to ensure consistency and dependencies. These formalisms may form the basis for automated software testing.

Another advantage of a formal model specification is the automated generation of application code. A programming language is a formalism where syntax and semantics are fully defined. This characteristic makes compilers possible, which among other things verify formal correctness. Another example of the benefits of this formalism is static code analysis tools, which can search source code to find bugs, enforce coding standards, find unreachable code, and analyze code complexity. Rigor and formality influence several dimensions and qualities of software products, including reliability, verifiability, maintainability, reusability, portability, understandability, and interoperability.

Rigorous, or even formal, software documentation can improve all of these advantages over informal documentation, which is very often ambiguous, inconsistent, and incomplete. Rigor and formality also apply to software processes.

### **21.3.2 Separation of Concerns**

*Separation of concerns* (Parnas 1972) enables us to deal with different aspects of a problem individually. This very important principle is applied to master the inherent complexity of software. When we separate concerns, the same system has several views, for example, its relationship with hardware, software behavior and its resources, space and time complexity, user interfaces, and interfaces with external systems.

Other concerns are related to the software development process and involve the development environment, team organization and structure, project scheduling, control procedures, or design strategies. Separation of concerns may be performed in terms of time, as in the project schedule or tasks; in terms of qualities, like efficiency and correctness; in views, as data flow, control flow, or activities that are synchronized; and in parts, as decomposition of a system into modules. Separation of concerns may also result in separation of responsibilities in dealing with separate issues. In principle, this would result in dividing the work on a complex problem into specific work assignments for different people with different skills. These work assignments could be, for example, optical systems design, machine vision algorithms, robot programming, database programming, application domain expertise, and project management.

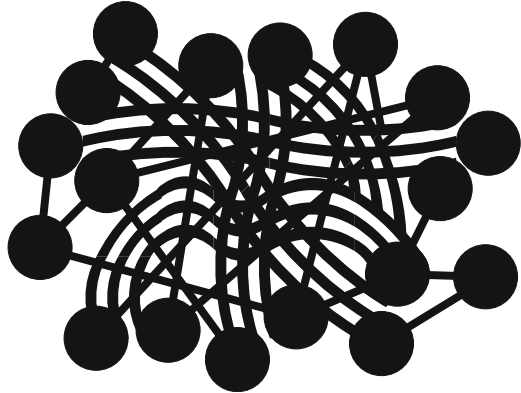
In an apparent disadvantage, separating concerns may result in the loss of some potential global optimizations, but in reality, by combining concerns we are likely to make wrong decisions because we are unable to overcome complexity. Concentrating on different aspects separately yields much better results, even at the expense of missing some global optimizations.

### **21.3.3 Modularity**

Modules are small parts extracted or abstracted from a complex system. Modularity is an important property of most engineering processes and products, for example, in optics, mechanics, and electronics. The main benefit of modularity is that it allows the principle of separation of concerns to be applied in two phases: first, when dealing with each module in isolation (bottom-up design) and second, when dealing with the overall characteristics of all modules and their relationships as a system (top-down design).

Modularity is not only a design principle, but it permeates the whole of software production. In particular, modularity pursues three important goals: decomposing a

**Fig. 21.3** Highly coupled structure



complex system into modules, composing a new system from existing modules, and understanding a system in pieces.

Decomposing a complex system into modules is just another example of the age-old technique of “Divide and Conquer” to solving complex problems. Composability of a system is based on a bottom-up construction of a complex system by using elementary components. Examples of this approach include microscopes, lasers, automobiles, airplanes, and computers. In software engineering, we aim to assemble new applications by taking modules from a library of reusable components and combining them to form the required new product. The capability of understanding each part of a system separately aids in the maintenance and modification of a system.

To achieve modular composability, decomposability, and understanding, modules must have two very important properties:

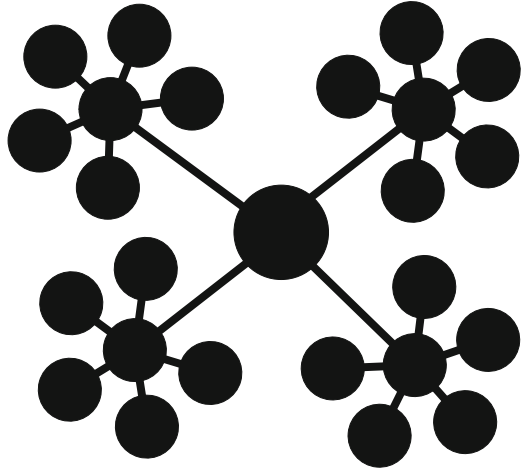
- High cohesion
- Low coupling

A module has high cohesion if all its elements are strongly related, for example, statements, procedures, and declarations, and they are together for a reasonable and well-defined specific purpose.

Coupling is determined by the relationship of an individual module with other modules in the same system. Coupling is a measure of the interdependence of two or more modules in the form of function calls or shared variables, for instance. If two modules depend on each other heavily, then we say that they are highly coupled. If this is the case, it will be difficult to analyze, understand, modify, test, or reuse them separately. Figure 21.3 attempts to illustrate this situation.

In contrast, modular structures with high cohesion and low coupling allow us to see modules as black boxes when the overall structure is described. The functionality of such kind of modules can be analyzed and described separately. This is another example of the principle of separation of concerns. This concept is exemplified in Fig. 21.4.

**Fig. 21.4** High cohesion, low coupling



### 21.3.4 Abstraction

Abstraction is a process to identify the important aspects of a phenomenon and ignore the details that are not relevant. What must be considered as important and be abstracted away and what should be considered as a detail to be ignored depend on the purpose of the abstraction. There may be different abstractions of the same reality, each providing a different view. All of these may be valid at the same time, but each serving a different, specific purpose.

For example, consider an electronic calculator. A useful abstraction for the owner is a description of the effects of the various buttons. For a service technician, it may be a box that can be opened to replace a battery. Other abstractions may be useful for understanding the activities needed to repair the unit.

Another example is the representation of electrical devices using mathematical equations. An equation represents a simplified, idealized model of the device. The mathematical model is an abstraction that ignores details concerning the manufacturing of each device, like process variations and tolerances, for example. All mathematical models are abstractions from reality, ignoring certain facts and concentrating on others that are deemed relevant. A model of a software system is an abstraction built on the software requirements specification. This model may be expressed at various levels of rigor and formality, including mathematical formalisms.

Programming languages are abstractions built on top of the hardware. These abstractions allow software developers to write programs at a “high level,” ignoring (in most cases) such details such as microprocessor type, number of bits, or addressing mechanisms. Computer programs are abstractions themselves, too. For example, a payroll software system provides the essence of the manual procedure, not its exact details.

Abstraction is an important principle that applies to both software products and processes.



### 21.3.5 *Anticipation of Change*

To ensure the ability of software to evolve is not straightforward, and it requires an explicit effort during the requirements analysis and design phases to anticipate how and where the changes are likely to occur. When potential changes are identified, the system must be designed in such a way that it will be prepared or make it easy to incorporate those changes when the time comes. Examples of these likely changes are multiple iterations of related products based on a product family platform. Anticipated modifications should be confined to restricted portions of the software, where the ripple effect caused by its insertion is well understood.

Many software systems are built when actual requirements are still not completely known, and later on, user feedback from the field provides the missing and necessary information for evolving the product. This is not the best way to build a product and certainly not the best way to design reusable software.

Reusability is also affected by anticipation of change. A software component is reusable if it can be directly incorporated into a new product. In practice, however, it is more likely that many reusable software components may have to be designed to accommodate slight changes in order to meet the requirements of a new application.

### 21.3.6 *Generality*

When we are asked to solve a problem, we should try to focus on the discovery of a more general problem that may include the problem at hand as a special case. It is possible that the newfound general problem may be not more complex than the original problem. The main advantage of this approach is that the solution to the generalized problem has more potential of being reused.

By generalizing the problem, we could possibly design a software module that is used more than once, perhaps providing slightly different services, rather than having several separate, specialized solutions. It is important to note, however, that in many cases, a general solution may be more expensive to develop and implement as compared to a specialized solution tailored to solve the original problem only.

Generalized solutions to some problems are exemplified by spreadsheets, databases, and word processors. These kinds of general-purpose solutions become off-the-shelf products and tend to be commoditized. This trend occurs in all branches of industry. For instance, in the early days of automobiles, it was possible to customize cars according to specific requirements of the customer, and “horseless carriages” were so expensive that only wealthy customers could afford them. Then, Henry Ford came along and decided that his customers could request cars of any color *as long as it is black* (standardized mass manufacturing). As the field

became more industrialized, customers could only choose from a catalog of models corresponding to prepackaged solutions offered by each manufacturer.

Nowadays, in an interesting return to the origins of artisan manufacturing, there is a marketing trend called “mass customization,” where the aim is to take advantage of advanced computer-aided manufacturing technologies to provide customized products to a large number of customers. This trend is a major motivation for the development of product family platforms, and it applies to software too.

### ***21.3.7 Incrementality***

Incrementality in software defines a development process that proceeds in a step-wise fashion, where each step is a small change or increment, as compared to a previous release of the system under development. Eventually, the system is completed after successive approximations to the full implementation of requirements. A major advantage of this approach is that development teams have a functional product at all times. Incrementality, as applied to software, means that the desired application is produced as the outcome of an evolutionary process.

Applying the principle of incrementality, we may add functions to the application being developed, one by one, starting from a prototype with minimum functionality, enough to make it useful, albeit incomplete, and thus continue with progressive iterations until the implementation is complete. Therefore, the early versions might emphasize user interfaces, reliability, and correctness, while later releases could focus on improving performance.

As with anticipation of change, an evolutionary process requires careful management of documents and code developed for the various versions.

## **21.4 Software Engineering Techniques**

This section introduces some basic concepts of software engineering techniques and terminology, necessary to comprehend the case study presented in Chap. 26.

Let us see how the fundamental ideas and software engineering principles introduced in the previous sections are applied to the design of reusable software architectures for a specific industry domain.

### ***21.4.1 Design for Change***

Design for change summarizes our approach to software design, and it is one of the driving ideas in our design process. We view software design as the decomposition

of a system into modules. A software design includes a description of what each module is intended to do and of the relationships among the modules. Such description is also called software architecture.

During the design activity, we must anticipate the changes that the system may undergo during its lifetime. Design for change promotes a design that is flexible, which in this case means that it is prepared to be modified easily. However, it is impossible to know everything about future changes with enough precision, and these may not be obvious. The ability to anticipate likely changes is, therefore, highly dependent on the previous experience of the software architect and the depth of understanding of the problem domain by the end user.

Design for change is very important, since changes to the system *will inevitably arise* afterwards. It is a common mistake to design a system for today's requirements, paying little attention to likely changes in the future life of the product. The consequence of this approach is that even a good original design may turn out to be extremely difficult to adapt to future requests for changes.

Frequently, in order to incorporate even seemingly minor changes, it results necessary to redesign and rewrite the whole application again. Another unfortunate consequence is that the original software structure is cluttered or patched, resulting in an application that is more difficult to maintain and less reliable.

The most common types of changes that a design should try to anticipate are change of algorithms, change of data representation, change of underlying abstract machine, change of peripheral devices, change in the social environment, or the evolution of program families. These are described below.

#### **21.4.1.1 Change of Algorithms**

As examples of changes in algorithms, we could mention image analysis algorithms in a machine vision application where we look for different image features or have to inspect a different product; new protocols for testing products; the motion sequences for a robot, in order to build new products; or writing faster algorithms for digital signal processing. Another example could be a toolbox of optimization algorithms where each member of the set uses a different approach for solving a generalized search problem with different levels of breadth, depth, or computational cost. For instance, solving a multivariable optimization problem using any of the following algorithms: genetic algorithms, particle swarms, simulated annealing, or Tabu search. As part of a system where all these algorithms are replaceable, all of them would comply with the same interface and would be directly replaceable.

#### **21.4.1.2 Change of Data Representation**

As an example of this type of change, we could mention that sometimes it is possible to improve the efficiency of an existing program by changing the structure used to represent data. For instance, changing the structure to represent a table from

a sequential array to a linked list can improve the efficiency of operations accessing the table. Another simple example is to add fields to (or deleting from) records, as more (or less) information is required to be stored in a file. If this implementation detail is hidden within a module (encapsulated), it makes the module a good candidate for reuse.

#### **21.4.1.3 Change of Abstract Machine**

Computer programs that we write run on top of an abstract machine that hides the details of the underlying hardware platform. Examples of abstract machines are the Java™ virtual machine, the Microsoft .NET Framework™, operating systems, and database engines.

Very often, it is necessary to modify our applications in order to be able to run on new versions of the operating systems, or we may want to use a different database engine in order to interact with different factory information systems.

#### **21.4.1.4 Change of Peripheral Devices**

As products evolve and mature, in many cases a whole ecosystem is developed around a successful product, such as the case of popular personal computers like Apple II™ or the IBM PC™. New peripherals continually arrive to the market, solving previously unmet (or unknown) needs. As an example, we may want to use a higher resolution camera or screen or use a different kind of industrial networking protocol. We may also want to use a faster laser scanner for our new application or perhaps to split an image-processing task between two processor boards instead of one.

#### **21.4.1.5 Change of Social Environment**

Change of social environment occurs when new products are deployed in foreign countries, which may have different languages, different alphabets, and different standard systems of units and measures, for example, the adaptation of user interfaces to languages like Spanish, Japanese, or Chinese and information display format for different countries, like temperature in Celsius or Fahrenheit degrees, date formats, decimal numbers, millimeters, or inches.

#### **21.4.1.6 Program Families**

In the case of software, the introduction of new products consists of building new versions of the same application, where every version constitutes an individual product, which might be sold as different “editions” (e.g., personal, professional,

and corporate editions). By designing all the members of a family jointly, rather than doing separate designs for each member, we avoid the cost of designing the common parts separately.

A simple example is an application that needs to be used by inexperienced users in their native language, as in the case of an operator in a factory overseas, for instance. Another example is a machine vision application that must run the same analysis algorithms with different types of cameras or at different image resolutions.

The more we stress commonality, the less work is done for each new version. This reduces the chances of inconsistencies and reduces the combined maintenance effort. Specific domain applications may be abstracted into an application framework, which allows the reuse of both design and actual code.

### 21.4.2 *Software Design*

Software design is basically the process of decomposing a complex system into a collection of individual modules that can be understood and implemented by a single person (top-down design), while maintaining a composite whole that is self-consistent and assembles these modules into a system using a logical structure (bottom-up design). The system's logical structure includes a clear and complete definition of the communication mechanisms that will enable the cooperation of these individual modules (module interfaces).

The principle of rigor and formality leads us to adopt formal notations for describing software design, for example, the Unified Modeling Language (UML). For an overview and introduction to the language, please refer to Fowler (2004), Rumbaugh, et al. (1998) and Booch, et al. (1998). Separation of concerns, modularity, and abstraction are applied to tackle the inherent complexity of designing software systems. Anticipation of change and incrementality will allow us to design systems that are able to evolve as requirements change. Anticipation of change and generality will enable us to develop families of products, *frameworks*, and other *middleware* components.

#### 21.4.2.1 **Top-Down Design**

The decomposition of a system into modules can be accomplished in several ways, as when designing top-down and bottom-up, for example. For top-down design, we can decompose a system into high-level subsystems first. Next, each subsystem is analyzed separately, and the procedure is iterated until the resulting modules are sufficiently small that a single person can implement it.

In order to arrive to optimal subsystem implementations, the use of Design Patterns is highly recommended (Gamma et al. 1995; Buschmann et al. 1996). Design patterns are object-oriented, language-independent proven design solutions

that are documented in a standardized format. These solutions help reuse design, rather than code.

In addition to modularity, rigor and formality are useful in the description of software architecture. The more precise the description, the easier it is to divide software development into separate tasks that can proceed in parallel, with little risk of inconsistencies. A very important feature of all modules should be to have high cohesion and low coupling.

#### 21.4.2.2 Bottom-Up Design

The bottom-up design strategy consists of defining modules that can be iteratively combined to form subsystems. For example, a module may be designed to provide an easy way of accessing a peripheral device, masking the low-level primitives provided by the device, like a robot or a laser scanner, for instance. Bottom-up design is the typical case where we are reusing modules from a library to build a new system.

#### 21.4.2.3 Module Interfaces

When designing a system, we want to divide the software into components such that the internal details of each component can be designed independently of the other components. If each component becomes a work assignment to a different programmer on a team, then each programmer should be able to work on a component with as little knowledge as possible about how the other members of the team are building their own components. This is another instance of the principle of separation of concerns.

During design, we must define how the interaction among modules actually takes place. The set of services that each module provides to its clients is called its *Interface*, and the way these services are accomplished by the module is called the module's *Implementation*.

A good interface is a key aspect of good design, because it supports the separation of concerns. The actual details of the implementation of services are the module's private secret and should not be visible through the interface. This is also known as *Information Hiding*.

The interface of a module describes exactly what the clients need to know in order to use its services, and it represents an abstraction of the module, as viewed by its clients. The designer who is in charge of designing a module *M* only needs to know the interfaces of the other modules used by *M* and may ignore their implementation. Module *M*'s interface may be viewed by the designer as his or her task description. The interface of module *M* may also be viewed as a *contract* between *M* and its clients.

The interface records all and only the facilities the designer in charge of *M* agrees to provide to other designers. Clients may depend on what is listed in the

interface, and therefore, as long as the interface remains the same, M may change internally without affecting its clients.

In most practical cases, interfaces describe computational resources, such as variables that are shared among modules or functions that perform some kind of operation. To maximize evolvability, the interface of a module should export the minimum possible amount of detail. Another goal is to hide low-level details and provide an abstract interface in order to make the design more understandable. These characteristics are a consequence of the principles of abstraction and separation of concerns.

We say that the changeable, hidden information becomes the secret of the module, or in object-oriented jargon, *encapsulated* within the module implementation.

#### 21.4.2.4 Abstract Objects

A significant number of modifications to software are due to changes in data representation. One very important type of encapsulation is hiding the details of data representation, isolating clients from internal changes in the module. Hidden data structures provide modules with a state. Modules that exhibit a state are called *Abstract Objects*.

A module that hides a data structure as a secret and exports functions that may be used to access the hidden data structure is an example of an abstract object. If for any reason the data structure had to change, the modification would be limited to the internal change of the algorithms that implement the access routines. In this case, client modules continue to use the same functions to access the hidden data, completely unaware of the existence of the change. Clients, therefore, do not need to change, as long as the interface remains the same.

#### 21.4.2.5 Abstract Data Types

A module encapsulating a data structure defines a *type*. We may also generate multiple instances of that type. Functions exported by such a module allow access to its hidden structure, providing a mechanism to manipulate instances of that type. Thus, an *abstract data type* is a module that exports a type, allows client modules to declare variables of that type, and performs operations on those variables through the type's exported operations.

### 21.4.3 Object-Oriented Design

Object-oriented design is a technique that pushes to the extreme a design approach based on abstract data types. An abstract data type is represented by a *class*, and a

class may be seen as a template for making objects. *Objects* are instances of a class. For example, people are instances of the “person” class. An object has *structure*; that is, it has attributes (properties) and behavior, which in turn, consists of the *operations* it carries out (also called functions or methods). Attributes and operations taken together are called *features*.

### 21.4.3.1 The Object Model

As objects in the *person* class, we all have the following attributes: height, weight, and age. We also perform these operations: eat, sleep, read, write, speak, and so forth. Object orientation goes beyond just attributes and behavior. It includes other aspects as well, namely, abstraction, encapsulation, inheritance, polymorphism, message sending, associations, and aggregation.

### 21.4.3.2 Abstraction

*“An Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide clearly defined conceptual boundaries from the perspective of the viewer”* (Booch 1991).

Different viewers focus on different essential characteristics of the same object. For example, a child may view a cat as a pet with a characteristic furry appearance, whereas for a veterinarian, it is a mammal with certain internal structure. The abstraction of an object becomes the definition of its interface. The interface must be kept to a minimum in order to make the module understandable and usable.

### 21.4.3.3 Encapsulation

*“Encapsulation is the process of hiding all the details of an object that do not contribute to its essential characteristics”* (Booch 1991).

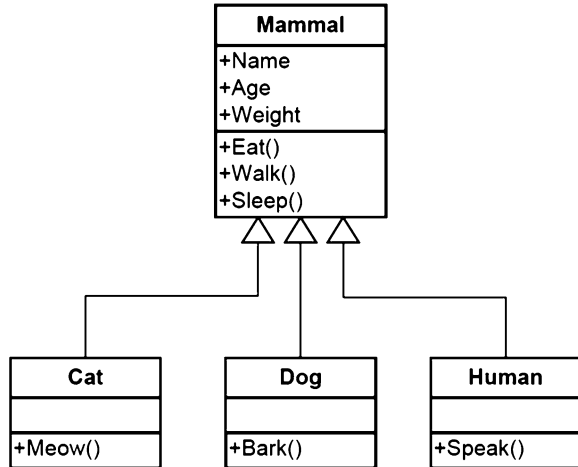
The explicit division of interface and implementation represents a clear separation of concerns: the implementation encapsulates details about which clients may not have knowledge or make assumptions. These are the “secrets” of an abstraction.

### 21.4.3.4 Inheritance

As mentioned before, objects are instances of a class. A class defines a category of objects, with certain attributes, operations, and a hidden data structure. When an object is instantiated, it has all the characteristics of its class, and we say then that the object *inherits* all the characteristics from its base class.



Fig. 21.5 Object inheritance



Inheritance is a powerful mechanism for reuse, since not only objects inherit from a class, but classes can inherit from other classes. For example, each of the classes *Cat*, *Dog*, and *Human* inherit all the characteristics of the class *Mammal*, as shown in Fig. 21.5.

In this case, we also say that *Cat*, *Dog*, and *Human* are subclasses of the *Mammal* class. The *Mammal* class, in turn, is called a *superclass* of *Cat*, *Dog*, and *Human*.

Inheritance, then, is a relationship among a hierarchy of classes, where lower classes in the hierarchy (*subclasses*) inherit the characteristics of higher classes in the hierarchy, also called *ancestors* or *superclasses*. There is no limit to this hierarchy, and thus, the *Mammal* class may inherit characteristics from the *Animal* class, which in turn inherits characteristics from the class *Living Being*, and so forth.

When a change is made to an ancestor class, the change is propagated to the whole hierarchy. Inheritance is one of the most powerful features of object-oriented languages.

### 21.4.3.5 Polymorphism

Polymorphism means “many forms.” With object-oriented programming languages, it is possible to provide many forms of behavior by providing different implementations of an inherited operation specified in an ancestor class. This way, different objects subclassed from the same ancestor may respond in different ways when this operation is invoked. Figure 21.6 illustrates this concept.

### 21.4.3.6 Message-Sending

Objects work together in a system. Objects communicate by sending and receiving messages. For example, an object sends another object a message to perform an

Fig. 21.6 Object polymorphism

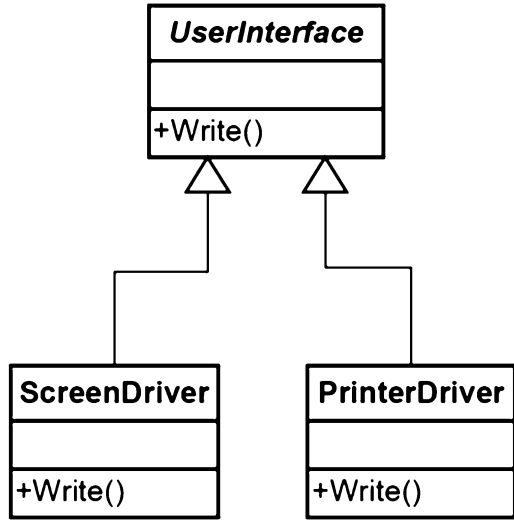
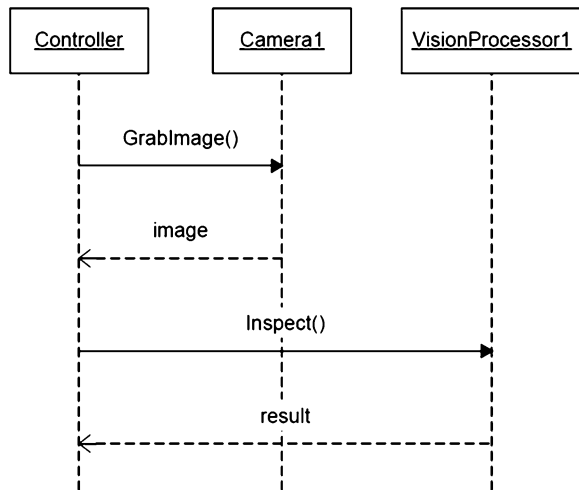


Fig. 21.7 Message passing between objects



operation. The receiver object recognizes the message and performs the operation and returns a result, as described by the sequence diagram shown in Fig. 21.7.

### 21.4.3.7 Associations

Associations represent relationships between objects (instances of classes). These associations express relationship concepts like *uses*, *extends*, *employee of*, *client of*,

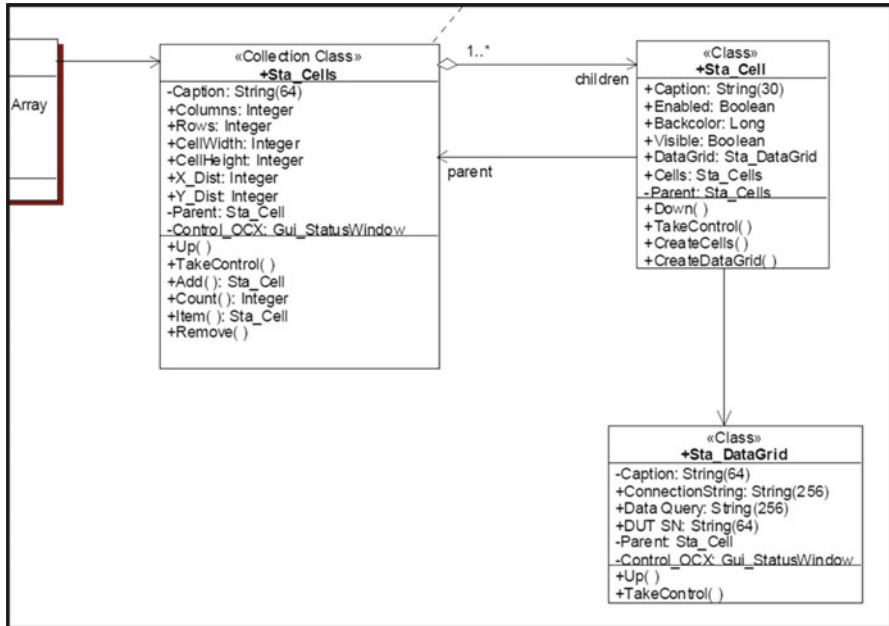


Fig. 21.8 Object association

and friends. Associations may be unidirectional (e.g., *parent of*) or bidirectional (e.g., *siblings*). Associations also have multiplicity, for example, *one-to-one*, *one-to-ten*, and *one-to-many*. In an association, each party may have a *role name* defined, for example, *sales rep*, *order*, and *line item*. Within the specification perspective, associations represent responsibilities of each object. Figure 21.8 shows some examples of class associations.

### 21.4.3.8 Aggregation

Aggregation is a special kind of relationship between objects. It may be thought of as the *part-of* relationship. An aggregation is a set of interrelated objects. For example, objects of type *memory*, *screen*, *cd\_rom*, *keyboard*, and *processor* are all aggregates that are part of an aggregate object of type *computer*.

An important characteristic of aggregations is that one or more of its components may be missing, while the aggregate object retains its identity. In the example above, the *cd\_rom* object might be removed, and the *computer* object would still be a computer. The set of aggregate objects makes up a single whole, as exemplified in Figs. 21.9 and 21.10.

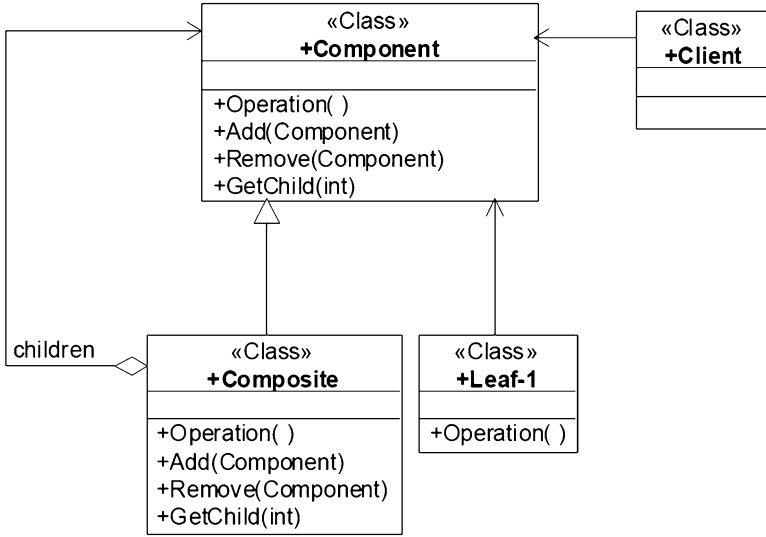


Fig. 21.9 Structure of an object aggregation

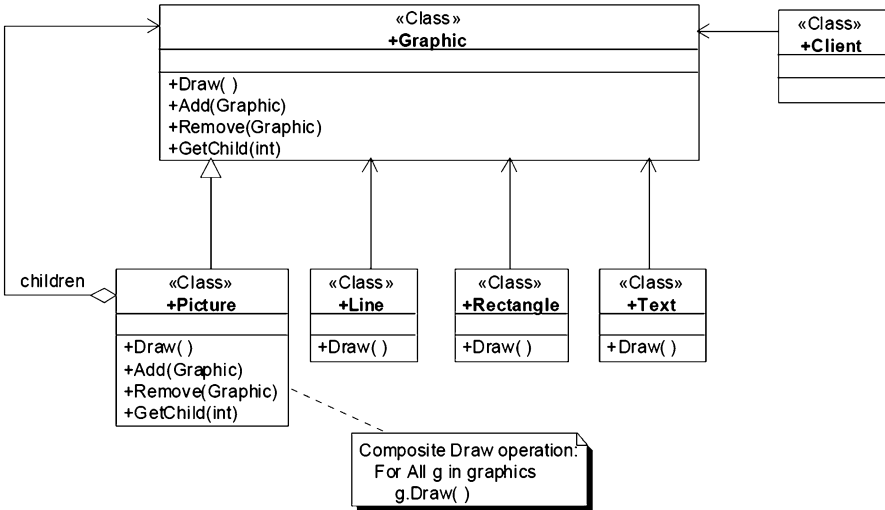


Fig. 21.10 Example of object aggregation

### 21.4.3.9 Composition

Composition is a stronger kind of aggregate relationship, where each and every one of its components is essential. With composition, the “part” object may belong to only one whole. For example, a cell phone keypad is composed of *key* objects, each

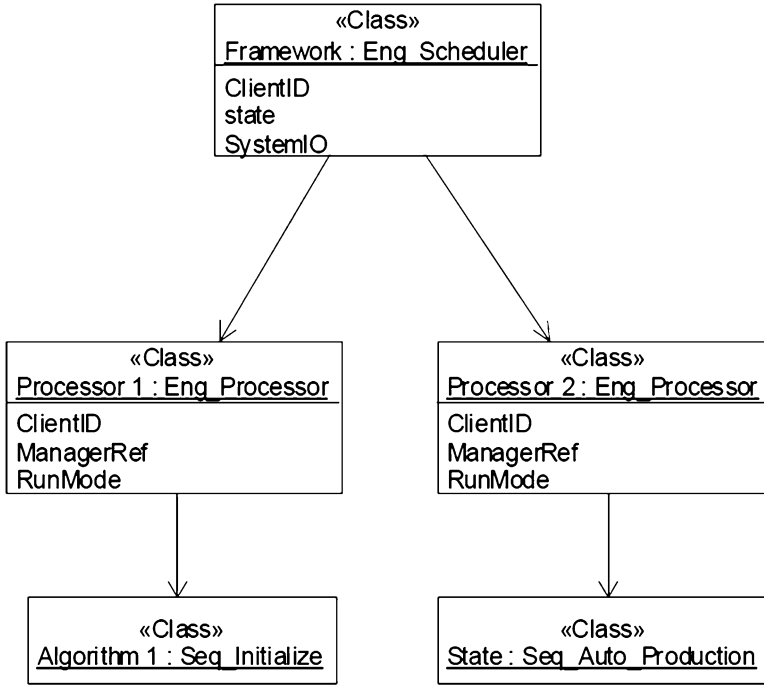


Fig. 21.11 Object composition

of them unique, even though all of them may be subclassed from the same abstract class. However, if one of them is missing, the composite object *keypad* is rendered useless.

Composition is particularly important in design patterns, which will be presented in the next section, and it is a powerful technique for software reuse. Complete systems may be constructed using object composition, where object relationships only exist at run-time. Figure 21.11 shows an example of this concept.

### 21.4.4 Software Reuse

The two most common techniques for reusing functionality in object-oriented systems are class inheritance and object composition. Class inheritance lets us define the implementation of one class in terms of another. Reuse by subclassing is often referred to as “White box reuse.”

Object composition is an alternative to class inheritance. New functionality is obtained by assembling or composing objects to get more complex functionality. Object composition requires that the objects being composed have well-defined interfaces. This style of reuse is called “Black box reuse.”

#### 21.4.4.1 Component-Based Software

Object composition enables a higher level of software reuse based on self-contained software components. Software systems made out of code components exhibit the highly desirable characteristics of high cohesion and low coupling.

Object composition is defined dynamically at run-time through objects acquiring references to other objects. Composition forces modules to respect each other's interfaces. Because objects are accessed solely through their interfaces, encapsulation is not broken. Any object can be replaced at run-time by another object as long as it has the same type.

Design patterns are proven design solutions based mainly on object composition at run-time. For details on design patterns, see Gamma et al. (1995) and object-oriented design (Booch 1991; Jacobson et al. 1992).

#### 21.4.4.2 Object-Oriented Application Frameworks

Object-oriented application frameworks embody the ultimate expression of software reuse. A framework is a set of cooperating classes that make up a reusable design for a specific class of software. In other words, frameworks are partially finished applications that address a specific domain.

A framework reuses architecture, design, and code, and they are customized to particular applications by creating application-specific subclasses of abstract classes from the framework. Thus, frameworks dictate the architecture of the new application, forcing fixed characteristics and assignments within the system, for example:

- Defines the overall structure
- Defines partitioning into classes and objects
- Key responsibilities of the objects
- How classes and objects collaborate
- Thread of control and execution

A framework predefines these design parameters so that software designers/implementers can concentrate on the specifics of the new application. In other words, the framework captures design decisions that are common to its application domain. Therefore, frameworks emphasize design reuse.

Reuse on this level leads to an inversion of control between the application and the software on which it is based. When you use a framework, you reuse the main body and write the application-specific code it calls. You will have to write operations (functions) with specific names and calling conventions, but that reduces the design decisions you have to make. Applications are built faster and have similar structures, making them easier to maintain and to appear more consistent to the user. All this convenience, however, comes with a price. If applications are hard to design and toolkits are harder, then frameworks are the hardest of all.

A framework designer gambles that one architecture will work for all applications in the domain. It is imperative to design the framework as flexible and extensible as possible. Furthermore, because applications are so dependent on the framework for their design, they are very sensitive to changes on the framework interfaces.

For more information on object-oriented application frameworks, see Brugali and Fayad (2002), Fayad et al. (2000), Johnson and Foote (1988), Roberts and Johnson (1997) and Fayad and Schmidt (1997).

## 21.5 Conclusions

This chapter introduced fundamental software engineering concepts, software qualities, and software engineering techniques necessary for a formal and disciplined approach to software design.

Object-oriented application frameworks are the software embodiment of a platform for a family of products, and to deal effectively with this complex software entity, it is necessary to master the concepts of software engineering principles, object-oriented software design, and design patterns, especially before embarking on the mission of designing one as a product family platform for a specific domain.

All of these concepts are quite abstract and will be more clearly understood after reading the software application case study presented in Chap. 26.

## References

- Booch G (1991) Object-oriented design with applications. Benjamin Cummins, Redwood City, CA
- Booch G, Rumbaugh J, Jacobson I (1998) The unified modeling language user guide. Addison-Wesley, Reading, MA
- Brugali D, Fayad M (2002) Distributed computing in robotics and automation. *IEEE Trans Robot Automat* 18(4):409–420
- Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M (1996) Pattern-oriented software architecture: a system of patterns. Wiley, Chichester, England
- Fayad ME, Schmidt D (1997) Object-oriented application frameworks. *Commun ACM* 40(10):32–38
- Fayad ME, Hamu DS, Brugali D (2000) Enterprise frameworks: characteristics, criteria and challenges. *Commun ACM* 43(10):39–46
- Fowler M (2004) UML distilled, 3rd edn. Addison-Wesley, Reading, MA
- Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, MA
- Ghezzi C, Jazayeri M, Mandrioli D (1991) Fundamentals of software engineering. Prentice Hall, Upper Saddle River, NJ
- Jacobson I, Christensson M, Jonsson P, Overgaard G (1992) Object-oriented software engineering: a use case driven approach. Addison-Wesley, Reading, MA
- Johnson RE, Foote B (1988) Designing reusable classes. *J Obj Orient Prog* 1(2):22–35

- Parnas DL (1972) On the criteria to be used in decomposing systems into modules. *Commun ACM* 15(12):1052–1058
- Roberts D, Johnson R (1997) Evolve frameworks into domain-specific languages. In: Martin RC, Riehle D, Buschmann F (eds) *Pattern languages of program design 3*. Addison-Wesley, Reading, MA
- Rumbaugh J, Jacobson I, Booch G (1998) *The unified modeling language reference manual*. Addison-Wesley, Reading, MA



# Chapter 22

## Considering Human Variability When Implementing Product Platforms

Christopher J. Garneau, Gopal Nadadur, and Matthew B. Parkinson

**Abstract** Design for Human Variability (DfHV) is the practice of designing artifacts, tasks, and environments that are robust to the variability in their users. Designs often incorporate adjustability and/or offer several sizes to account for the different requirements of the target user population. There are several situations where DfHV can provide platforming opportunities that might otherwise be overlooked. This chapter provides a brief introduction to DfHV, outlines some basic techniques, and provides a description of scenarios where platforming and modularity might be a good approach.

### 22.1 Introduction

Product families and platforms are designed to efficiently provide the required variety in a product portfolio. This is usually defined through engineering and market research—the former provides information about usage conditions and the latter identifies the needs of the business, users, and any regulatory requirements with which compliance is required. Many of these requirements vary across global regions, markets, and consumers. Engineers are trained to consider many kinds of

---

C.J. Garneau  
OPEN Design Lab, The Pennsylvania State University, University Park,  
PA 16802, USA

U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, USA

G. Nadadur  
OPEN Design Lab, The Pennsylvania State University, University Park,  
PA 16802, USA

M.B. Parkinson (✉)  
OPEN Design Lab Engineering Design, Mechanical Engineering, and Industrial Engineering,  
The Pennsylvania State University, University Park, PA 16802, USA  
e-mail: [parkinson@psu.edu](mailto:parkinson@psu.edu)

variability. For example, there are robust design methodologies for incorporating an understanding of the variability in materials and manufacturing processes associated with producing a product. In the artifacts and environments with which humans interact, one of the largest sources of variability is the users themselves. Because users can often adapt to make a design work, their needs are sometimes considered less important than the hard engineering constraints that are part of every project. Design for Human Variability (DfHV) is a D-f-X capability in which artifacts, tasks, and environments are designed to be robust to the full variability in their users. This chapter will focus primarily on body size and shape, but the techniques presented apply to other types of DfHV (e.g., physiological, cognitive, or perceptual).

Although the ideal is a truly *universal* design that accommodates everyone equally well, practical limitations such as cost, development time, and conflicting user requirements make this impossible. Instead, the designer or ergonomist selects a design topology and dimensionally optimizes the product, or environment, with the objective of providing some level of accommodation for its target users (Roe 1993). This is often achieved through adjustability and/or sizing. The use of modularity and product platforms can assist designers in using these approaches in cost-effective and strategic ways. The following sections explain some basic principles of DfHV, outline available data and other resources, and give examples of common practice for sizing and adjustability. The chapter concludes with a description of some specific opportunities for applying platforming and modularity—“How does one recognize a DfHV problem where platforms might be particularly appropriate?”

## 22.2 Basic DfHV Principles

Design for Human Variability combines rigorous design methods (e.g., optimization or robust design) with statistical modeling and ergonomics/human factors to improve safety, fit, performance, or other measures of accommodation of broad populations. *Accommodation* is the condition wherein a user can interact with a device in a preferred or safe way. Performance is commonly expressed in terms of the percentage of potential user populations accommodated. The application of design automation tools facilitates the simultaneous consideration of many attributes of the target user population (such as anthropometry, capability, and age) as well as other aspects of designs (Parkinson et al. 2007; Michalek et al. 2006; Zou and Mahadevan 2006; Van der Vegte and Horvath 2006; Osteras et al. 2006).

Properly accounting for the relevant variability in a target user population is the first step in a quantitative DfHV analysis. This includes identifying the target population (based on factors such as age, race, or demographics) and quantifying the variability within. The quantification can be achieved through experiments, appropriate databases (e.g., US Centers for Disease Control and Prevention 2010; Gordon et al. 1989), and estimation based on models of relationships in the measures.

Many traditional approaches to solving a DfHV problem involve the use of boundary manikins (Sundin and Ortegren 2006; Bittner 2000). In this approach, users at the extremes of the expected variability are used to assess candidate designs. The expectation is that if the design works for these boundary cases, the remaining users will be accommodated. Other approaches consider only an average user. As data quantifying variability have become more prevalent and computing tools more powerful, *virtual fitting trials* (Colombo and Cugini 2005; Garneau and Parkinson 2011) are increasingly used. Populations of hundreds or thousands of virtual users represent the diversity in anthropometry and preference of the target user population. The interaction of the population with a candidate design is then predicted, and the performance of the design for the entire population is assessed. The underlying models may be physics-based or statistical models involving an experiment with a prototype and a small number of representative users. A virtual fitting trial method facilitates the use of optimization methodologies (Parkinson and Reed 2006b). In particular, virtual fitting permits quantitative assessment of accommodation by offering the ability to check whether each virtual person is accommodated or disaccommodated by the variants that comprise the product portfolio.

## 22.3 Quantifying the Variability

Whether using boundary manikins, designing for some hypothetical average user, or using the more rigorous virtual fitting trials, estimates of the relevant variability within the target user population are needed. These can include variability in body size and shape, capability, and preference, among other factors. While an experimental study is typically required to estimate preference, there are many published estimates of anthropometry and capability. When the size and shape data for the specific population of interest are unavailable, techniques have been developed for obtaining estimates. These usually use statistical relationships in known data to infer or synthesize values in a new population. Biomechanics models can also be useful. This section focuses on variability in body size for three main reasons: (1) it is a reasonable proxy for the many types of variability in a population that affect physical interaction between user and device, (2) it is straightforward to collect during user trials, and (3) detailed data are available for many populations around the world.

### 22.3.1 Databases

Comprehensive anthropometric data are available for user populations around the world. Examples of these databases are Germany's Mikrozensus (Statistisches Bundesamt Deutschland 2004); India's survey of agricultural workers (Gite et al. 2009); Japan's Human Engineering for Quality of Life (Research Institute of

Human Engineering for Quality Life 1997); England's Health Survey (NHS Information Center 2009); China's Human Dimensions of Chinese Adults, GB 10000-88 (The State Bureau of Technical Supervision 1989); ANSUR (Gordon et al. 1989), which is a survey of the US Army in the late 1980s; and NHANES (US Centers for Disease Control and Prevention 2010), which is a continuous survey of US civilians. These databases contain some number of measures (e.g., stature, sitting height, arm length, and so forth) for each person sampled in the survey and also usually provide summary statistics and appropriate statistical weights.

There are three important caveats to consider in the use of these data: (1) anthropometric databases are not available for every user population a designer will encounter, (2) a given design may target a population with a different gender or racial/ethnic composition than available data, and (3) existing databases may lack data about certain body measures that are critical to the design task. Methods have been developed to synthesize anthropometric data in order to remedy these concerns.

Unlike anthropometry, large databases of capability and preference are simply not available. Although gathering data on body size and shape is expensive and time consuming, the problem is a finite one. Using a combination of traditional techniques (i.e., scales, calipers, and anthropometers) and scanners, extensive data can be gathered relatively quickly. Capability and preference, however, vary within individuals in an infinite number of ways. Strength, for example, varies with posture and the frequency with which the task is conducted. An individual's ability to maintain balance depends on whether they are standing or seated, have their eyes open or closed, and on what footwear they are wearing. And the conditions for measuring preference are even more specific. There are some resources available where the specific situation of interest has been investigated. These are primarily published in journals and technical reports. Some suggestions include the journals *Ergonomics*, *Biomechanics*, *Clinical Biomechanics*, *The International Journal of Industrial Ergonomics*, *Human Factors*, *Engineering Design*, and ASME's *Journal of Mechanical Design* and technical reports from *The University of Michigan Transportation Research Institute*, *SAE International*, and *JD Power and Associates*. In general, however, the designer will likely need to conduct their own study.

### 22.3.2 *Synthesizing Anthropometry*

An alternative to using actual anthropometric data is to use accurately estimated data. Accordingly, *anthropometry synthesis* is a key component of methodologies to design for human variability; such methods are usually applied when a representative database is unavailable for the target user population.

Traditional anthropometry synthesis techniques include those based on the use of proportionality constants and linear regression models. Proportionality constants are average ratios of various body measures to stature (Drillis and Conti 1966).

Given the stature data for a target population, the required body measure ( $\mathbf{Y}_i$ ) may be obtained by multiplying stature ( $\mathbf{S}$ ) with the appropriate proportionality constant ( $\mathbf{pc}_i$ ):

$$\mathbf{Y}_i = \mathbf{pc}_i \cdot \mathbf{S}. \quad (22.1)$$

While simple and easy to apply, this method is inherently flawed in many respects (Fromuth and Parkinson 2008), with the most fundamental error being the incorrect assumption that body proportions are constant across individuals.

Alternative anthropometry or posture prediction techniques (e.g., Flannagan et al. 1998; You and Ryu 2005) involve the formulation of regression equations between the data to be estimated ( $\mathbf{Y}$ ), which may be either relevant anthropometry or postures, and the chosen predictors, which are typically stature and/or BMI (body mass index, a measure of weight for stature):

$$\mathbf{Y}_i = \mathbf{a}_i \cdot \mathbf{stature} + \mathbf{b}_i \cdot \mathbf{BMI} + \mathbf{c}_i \quad (22.2)$$

where  $i$  is the number of required body measures or postures;  $\mathbf{a}$  and  $\mathbf{b}$  are regression coefficients; and  $\mathbf{c}$  is the regression constant. These regression relations can be based on data from existing databases or newly conducted surveys or experiments. These relations can be extrapolated to the target user population by using data about the predictors for the target population in the regression equations. Stature and BMI are effective predictors for several reasons: (1) they are strongly correlated with measures of length and breadth, respectively; (2) they are easily measured and readily available for a number of populations; and (3) they are uncorrelated with each other ( $R^2 \approx 0.02$ ).

There are three main drawbacks associated with this type of regression approach: (1) the incorrect assumption that people with the same values of predictors (e.g., stature) will also always have the same values of the estimated variable (e.g., leg length), (2) the lack of accuracy in simulating the distributions of anthropometry in the upper and lower tails, which are key to making accurate and efficient design decisions, and (3) the inability to model the component of user variability that is uncorrelated to the chosen predictors.

Incorporating residual variance into the regression model overcomes the limitations of the traditional linear regression models (Nadadur and Parkinson 2010) for this application. Residual variance is a statistical measure that quantifies the amount of variability in a regression model that is uncorrelated to the chosen predictors. The method reintroduces this measure in the form of a stochastic term in the standard regression equation:

$$\mathbf{Y}_i = \mathbf{a}_i \cdot \mathbf{stature} + \mathbf{b}_i \cdot \mathbf{BMI} + \mathbf{c}_i + \mathbf{N}_i(0, \mathbf{s}_i^2) \quad (22.3)$$

where  $\mathbf{N}(0, \mathbf{s}^2)$  is a normally-distributed stochastic term with mean 0 and variance  $\mathbf{s}^2$ , which is the residual variance of regression.

This method yields accurate estimates of data even in the upper and lower tails of distributions. This is an important improvement over traditional regression because

design decisions are usually made in the tails of anthropometric distributions, and it is the users in these upper and lower tails that are most likely to be affected by the decisions (Haslegrave 1986). Having accurate data in the tails is therefore key to making well-informed design decisions in developing the product portfolio.

## 22.4 Designing for the Variability

There are two main strategies for spatially optimizing a product so that it is comfortable or safe for its human users: (1) specifying a certain amount of *adjustability* on one or more dimensions of the same variant of the product, or (2) specifying *multiple size variants* of the product. While permitting adjustability is often simpler to design, it also introduces complexity and cost that may be inappropriate for some applications (e.g., apparel). However, specifying multiple sizes of a product also introduces design challenges. For instance, the outcome may be sensitive to design specifications (e.g., average versus minimum performance optimization); sizes may be distributed in various ways (e.g., equal people per size versus equal spacing across the variability in dimensions); and assessing the accommodation of a target population for size variants poses a greater challenge than for adjustable products.

These two strategies are discussed in the context of two design problems: bicycles and tool handles. The problems also demonstrate two different approaches to modeling user needs. The first (bicycle) uses data from an experimental study with participants representative of the target user population. The second (tool handle) uses equations from biomechanics and literature. The platforming of power tools is a well-known example within the product platform community (Lehnerd 1987; Meyer and Lehnerd 1997). Here it is examined differently, where the size and shape of the user provide platforming opportunities.

### 22.4.1 Specifying Adjustability

Adjustability is often specified in terms of required lower and upper limits to the adjustable product dimension under investigation. There are many recommended tools and practices that use anthropometry to prescribe adjustability (HFES 300 Committee 2004; SAE International 2006). Methods using only anthropometry include “manikin” approaches (Bittner 2000; Diffrient et al. 1981; UGS 2007) and population model approaches (Roe 1993). So-called hybrid methods using attributes of both manikins and population models have been shown to be an improvement on using one or the other individually (Reed and Flannagan 2000; Garneau and Parkinson 2011).

Recent research has explored including a “preference” component, which considers variability not predicted by body dimensions (Reed et al. 2000;

Parkinson et al. 2007; Parkinson and Reed 2006b). In particular, Garneau and Parkinson (2007) outline a method for determining the range of adjustability for a single size, continuously adjustable artifact including preference effects. Such a hybrid method including preference is based on the work of Parkinson et al. (2007) and Parkinson and Reed (2006a).

The first step for implementing a hybrid method including preference is to obtain experimental data from a sample group using a prototype similar to the product being designed. These data must include some anthropometric measure from the sample group (often stature) as well as a corresponding preferred device configuration. Next, linear regression is performed to relate the user-selected setting to a body dimension such as stature for the entire test population. Both the equation of the regression line and a measure of scatter (root-mean-square error, RMSE) must be retained. The parameters of regression are then used to construct a “virtual population” of a large size (e.g., 1,000 members). Each member of this virtual sample is assigned a stature randomly drawn from a representative database (e.g., NHANES or ANSUR). A preferred setting for each member is determined using the results of the regression, including a stochastic component calculated from the residual variance. The adjustability required to achieve the desired level of accommodation (e.g., 95 %) is determined by taking the maximum and minimum selections of the appropriate portion (e.g., the central 95 %) of virtual users. This hybrid method is utilized in the examples in this chapter.

### 22.4.2 *Specifying Sizes*

It can be preferable to offer a number of sizes to accommodate a range of body size and shape, capability, or preference. This can be done in conjunction with, or exclusive of, adjustability. Footwear, for example, comes in fixed, nonadjustable lengths. Bicycles, on the other hand, come in multiple sizes, but frequently have adjustable seats and handlebars.

One approach to identifying the sizes in which an artifact should be produced is to consider each size to be a variant within the product family. Although many recent DfHV studies have focused on optimally allocating product adjustability (Parkinson and Reed 2006b; Nadadur and Parkinson 2009; Garneau and Parkinson 2011), few have rigorously considered the optimization of product dimensions across multiple discrete sizes. McCulloch et al. (1998) describe a method of using optimization to improve fit with multiple sizes of apparel. Specification of sizes is also often guided by meeting certain industry specifications or manufacturing tolerance. Tryfos (1985, 1986) confirm this observation and offer another application for optimization to minimize garment discomfort in order to maximize sales. HFES 300 Committee (2004) provides one example of specifying sizes of military gloves in which the number of sizes is determined based on a minimum manufacturing tolerance. Nearly every application of size specification divides the dimensions of sizes evenly between the extremes, but this may not be desirable.

Aug. 20, 1929.

C. F. BRANNOCK  
FOOT MEASURING INSTRUMENT  
Filed Sept. 17, 1927

1,725,334

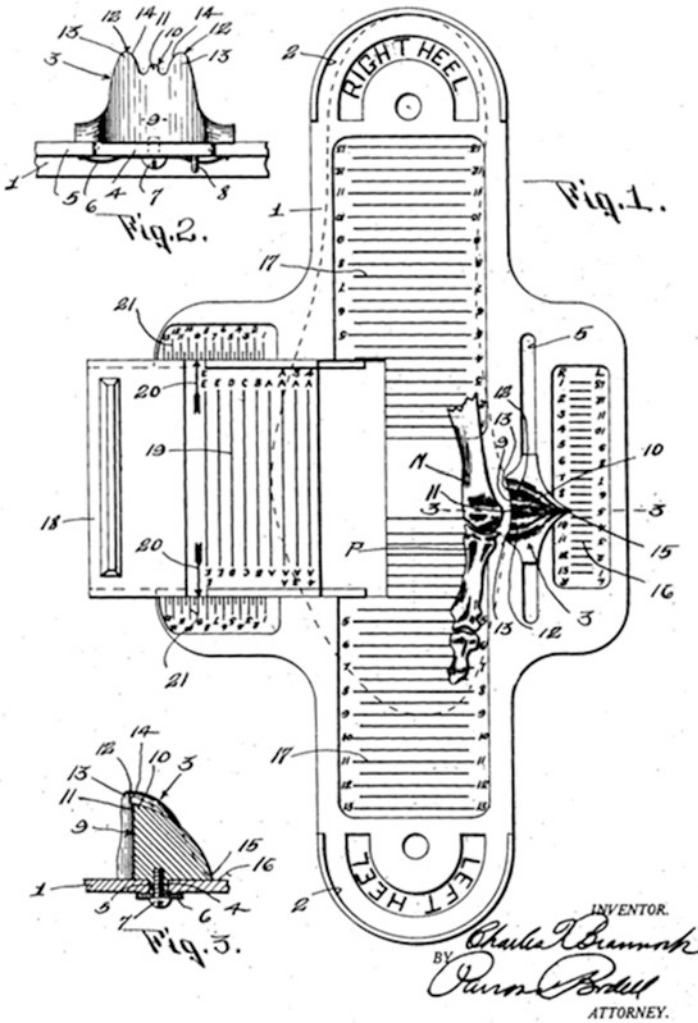


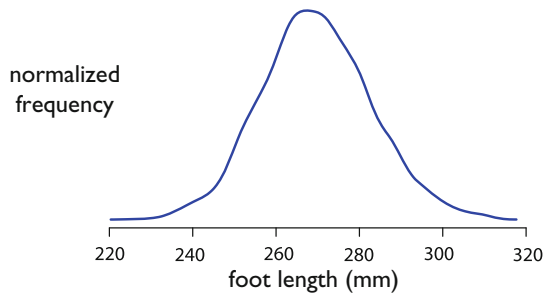
Fig. 22.1 An image from the patent application for the Brannock foot measurement device. Patent #1725334

### 22.4.2.1 Sizing for Equal Variability

The most common approach to the specification of sizes is to evenly allocate the variability across some specified number of sizes. For example, as can be seen when observing the markings on the Brannock Device (Fig. 22.1) used to estimate



**Fig. 22.2** Distribution of foot length within the male ANSUR population



appropriate sizes for footwear, the increments between each size are in fixed increments. Whether an individual’s foot is small, average, or large, a shoe that fits well on length can usually be found. However, the number of users accommodated by the central sizes will be much greater than those at the extremes.

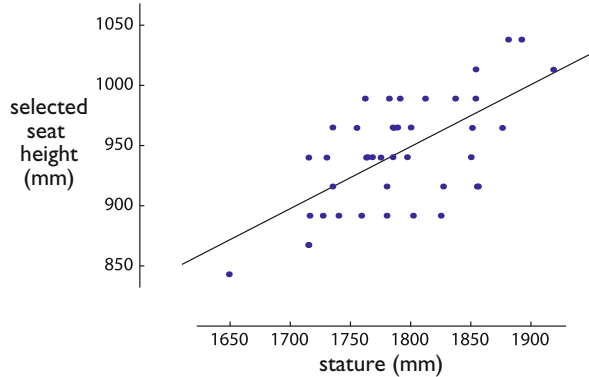
#### 22.4.2.2 Sizing for Equal Accommodation

The sizing for equal variability scheme described above results in an unequal demand for the product variants. As can be seen in Fig. 22.2, foot length is approximately normally distributed. Since there are substantially more individuals near the mean than in the tails of the distribution, one would expect that the demand for shoes of “average sizes” would be much greater than at the extremes. This creates a challenge at all levels of the supply chain. Consumers with feet of average size may find that their size is sold out, while those with large or small feet may find that their size was never even ordered. Vendors must stock a full range of sizes, but risk having their large and small inventory left over. Manufacturers must make unequal numbers of the different variants, causing problems in associated processes, materials, and tooling, and making it more difficult to leverage economies of scale.

The sizing for equal accommodation strategy mitigates these issues by targeting the variants such that each faces a theoretically equivalent demand. This is demonstrated in the footwear example by the omission of half-sizes (e.g., the larger men’s sizes might be 10.5, 11, 11.5, 12, 13, 14). By increasing the size increment in the larger sizes, the demand for a given size is artificially inflated (i.e., doubled). Although this is beneficial for vendors and manufacturers, it can decrease performance for the consumer.

Since this results in designs that must accommodate different amounts of variability, adjustability can (when appropriate) be incorporated to make up the difference. This is an interesting opportunity for platforming since key aspects of a design may simply be scaled across the sizes, while others—such as the amount of adjustability required to achieve the required accommodation—must be changed (variants near the mean require less adjustability to accommodate their assigned users than those nearer to the extremes).

**Fig. 22.3** The selections of the 42-member sample used in Garneau and Parkinson (2007). The parameters of a linear regression and the resulting line are also shown



### 22.4.3 Example 1: Bicycle Saddle Height

#### 22.4.3.1 Background

This analysis relies on the experimental data detailed in Garneau and Parkinson (2007); see that paper for information on the experimental method and results (only a summary is provided here). The device to be designed is an upright exercise cycle, and the target population is adult males. Therefore, the prototype is a typical upright exercise cycle, and the sample group was a set of 42 male students at Penn State University. The metric of interest is the minimum saddle height and its range of adjustability, which is used to determine an optimal number of sizes (with respect to cost). Ninety-five percent accommodation of the target population is sought. It is important to note that this analysis is provided as a simple case study only and is not intended to be a guide for bicycle saddle height design. Figure 22.3 shows the selections of the 42 sample users and the associated linear regression.

Linear regression is performed using the selected saddle height and stature for the sample to create a saddle height ( $H_{ground}$ ) preference model that incorporates residual variance:

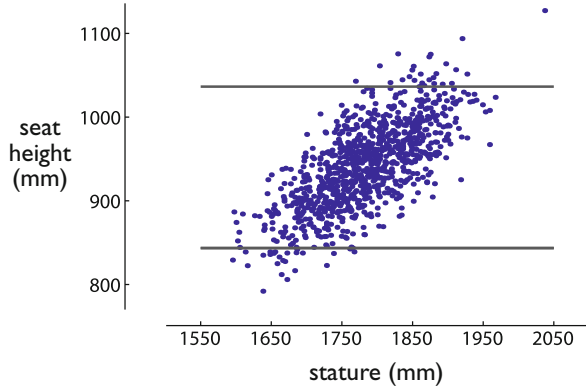
$$H_{ground} = 0.476S + 98.9 + N(0, 38.8) \quad (22.4)$$

where  $N$  is a normal distribution with a mean of 0 and a standard deviation of 38.8 (RMSE of the regression).

#### 22.4.3.2 Adjustability

The preferred saddle height, including a component indicating how their preference deviates from the mean, is calculated for each virtual user using the hybrid method and the regression equation (Eq. 22.4) outlined above. The required stature input was obtained by randomly sampling 1,000 individuals from the male ANSUR

**Fig. 22.4** The preferred seat locations of the 1,000-member virtual population. The central 95 % (950) users are noted



population. To achieve 95 % accommodation, saddle height selections at the 2.5th and 97.5th percentile give the low and high adjustment limits. This yields 192 mm of adjustability, as shown in Fig. 22.4.

Since the correlated body dimension, stature, is a measure of length, it is approximately normally distributed. Similarly, the preference related to anthropometry is also approximately normally distributed. As such, adjustability is best allocated by centering it on the mean preferred height. This provides the most accommodation for the smallest amount of adjustability. Not every scenario is like that. In some cases the distributions are skewed, and in others accommodation is limited on one end. For example, when specifying seat width, any individual with hip breadth smaller than the distance between the armrests fits without difficulty. For this situation, the lower 95 % of the population would provide the most efficient use of resources.

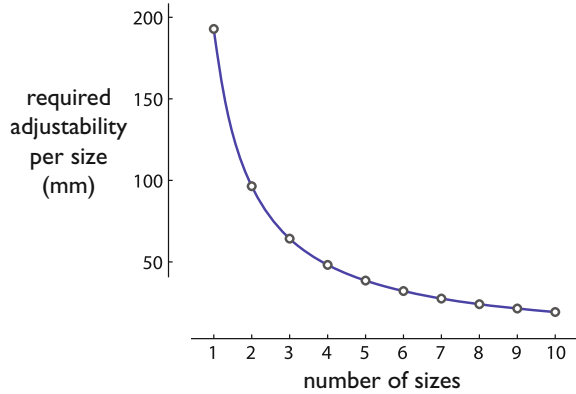
### 22.4.3.3 Sizing

One approach to identifying the size-related variants in a platform is to minimize cost across the platform. At its most basic, cost for an adjustable product with multiple sizes is proportional to the number of separate sizes, the quantity of each size produced, and the amount of adjustability within each size. Therefore, total cost may be broken into four components: fixed costs, the cost of offering a certain number of sizes, the cost of producing a given quantity of each size, and the cost of providing a certain amount of adjustability within each size. This may be represented mathematically as

$$\text{Cost}(n, q_i, \Delta X_i) = A + Bn + \sum_{i=1}^n \{C_i q_i + D q_i f(\Delta X_i)\} \quad (22.5)$$

where  $n$  = number of size variants,  $q_i$  = quantity of each size variant produced,  $\Delta X_i$  = amount of adjustability in each size variant, and  $f(\Delta X_i)$  = function relating adjustability to cost.

**Fig. 22.5** Required adjustability per size versus number of sizes for the exercise cycle example with each size having equal adjustability



The following constraints must also be satisfied:

$$\sum_{i=1}^n q_i = q \tag{22.6}$$

$$\sum_{i=1}^n \Delta X_i = \Delta X \tag{22.7}$$

where  $q$  = total quantity of all product variants produced and  $\Delta X$  = total overall adjustable range.

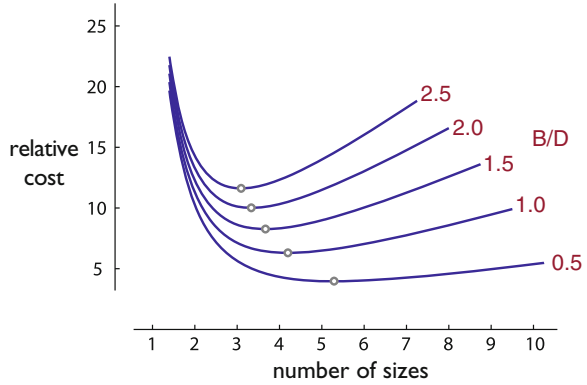
The constants in Eq. (22.5) are defined as follows:  $A$  represents fixed costs such as marketing and capital expenses.  $B$  is the cost of offering a size variant. This might include the cost of storing a size variant and reconfiguring machinery to produce different sizes, for example.  $C_i$  is the manufacturing cost per product for each size variant. In general,  $C_i$  may be different for each size and may be a function of the quantity of that size produced.  $D$  is the incremental cost of adjustability in each size variant. Each constant must be determined on a case-by-case basis.

#### 22.4.3.4 Sizing for Equal Variability

When the sizes are evenly distributed over the adjustable range, the required adjustability per size,  $\Delta X_i$ , is the same for all sizes and is given by  $\Delta X/n$ . Figure 22.5 demonstrates the nature of this relationship for the exercise cycle in which  $\Delta X = 192$  mm. An even distribution of sizes simplifies Eq. (22.5), since  $\Delta X/n$  may be substituted for  $\Delta X_i$ .

Two more assumptions make a simpler form of Eq. (22.5) possible. First,  $C_i$  is assumed to be the same across all size variants and independent of the quantity of each variant produced. Combined with the constraint of Eq. (22.6), this allows the cost of production to become  $Cq$  when summed across all sizes. Second, a function

**Fig. 22.6** Relative cost as a function of the number of size variants and the constant B/D for the exercise cycle example with sizes of equal adjustability. Various values for this constant are given to the right of the curves. The point of minimum cost is also shown on each curve



is assumed for  $f(\Delta X_i)$ . In general, any function appropriate to the application may be used. Here a nonlinear form will be assumed such that  $f(\Delta X_i)$  becomes  $(\Delta X/n)^2$ . This models the nonlinear increase in cost with increase in adjustability. Note that in the case of evenly distributed sizes, although  $\Delta X_i$  is the same for all sizes,  $q_i$  is not.

The cost objective function is now only a function of the number of size variants, if a certain total quantity  $q$  and amount of overall adjustability  $\Delta X$  are assumed. Formatting this objective function in the standard way gives

$$\min \text{Cost}(n) = A + Bn + Cq + Dq \left( \frac{\Delta X}{n} \right)^2 \tag{22.8}$$

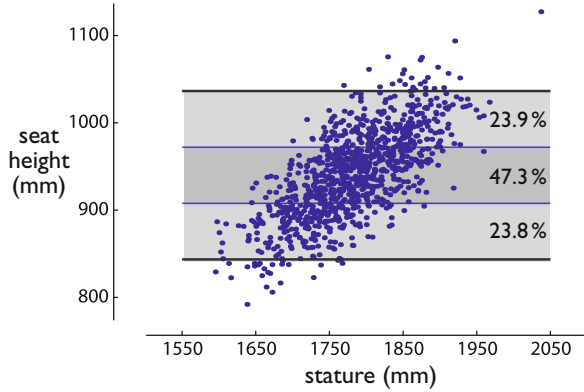
Neglecting  $A + Cq$  (since it is a constant for any number of size variants), a relative cost function may be formatted as

$$\min \text{Cost}(n) = \frac{B}{D}n + q \left( \frac{\Delta X}{n} \right)^2 \tag{22.9}$$

From this equation, one can see that an increase in the number of size variants increases production costs but decreases the cost of adjustability (since each size requires less adjustability). Therefore,  $B/D$  may be thought of as a penalty for adding additional sizes, or as the degree to which an increase in the number of sizes increases cost over an increase in the amount of adjustability per size.

Figure 22.6 shows cost curves in the case where the  $n$  sizes are divided evenly,  $q = 1,000$ , and  $\Delta X = 0.192$  m. Fixed costs are neglected, and so the curves represent relative costs for various values of  $B/D$ . The optimum number of size variants for each curve is given by the minimum cost. Suppose the value of  $B/D$  for the exercise cycle is determined to be about 2.5. Then the optimum number of size variants to minimize cost is about 3, determined by inspecting Fig. 22.6 or by using an unconstrained optimization algorithm. Notice that an increasing value of  $B/D$

**Fig. 22.7** The size limits for an evenly spaced sizing scheme are shown along with the 1,000-member virtual sample and the accommodation for each size variant. Note the varying levels of accommodation for each variant



means that offering more size variants becomes increasingly expensive over adding more adjustability per variant. Therefore, the optimum number of size variants decreases, as shown in the plot.

If the size limits of the exercise cycle are evenly distributed as shown in Fig. 22.7, then about 47 % are targeted by the middle band, and roughly 24 % are targeted by each of the outer bands. Therefore,  $\Delta X_{1,2,3} = 64$  mm,  $q_1 = q_3 = 0.24q$ , and  $q_2 = 0.47q$ .

### 22.4.3.5 Sizing for Equal Accommodation

When sizing for equal accommodation, an equal number of individuals is accommodated by each product variant. Since demand for each variant is expected to be the same,  $q_i$  is the same for all sizes and is equal to  $q/n$ . This requires that the adjustability for each size,  $\Delta X_i$ , is different across sizes. Assuming  $C_i$  is a constant across size variants as in the last case, the cost objective function then becomes:

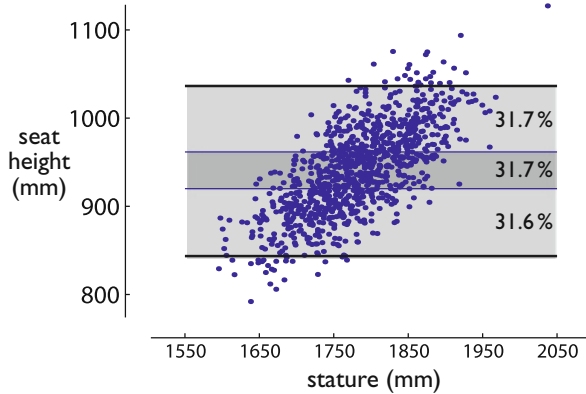
$$\min \text{Cost}(n, \Delta X_i) = A + Bn + Cq + D \left(\frac{q}{n}\right) \sum_{i=1}^n f(\Delta X_i) \tag{22.10}$$

$$\text{subject to } \sum_{i=1}^n \Delta X_i = \Delta X \tag{22.11}$$

Equation (22.10) is easily simplified only if  $f(\Delta X_i)$  is a linear function, i.e. the cost of adjustability is linearly related to cost. Therefore, a simplified cost objective function similar to Eq. (22.8) is not possible for a quadratic  $f(\Delta X_i)$ . If  $f(\Delta X_i)$  is linear, then optimizing cost with respect to sizes for evenly or unevenly distributed sizes yields the same results.

If the size limits are as prescribed by the nonlinear scheme of Fig. 22.8, then the proportion of the target population accommodated by each size variant will be

**Fig. 22.8** The size limits for a uniform accommodation sizing scheme are shown along with the 1,000-member virtual sample and the accommodation for each size variant. Note the equal levels of accommodation but different adjustable range per variant



evenly distributed. Therefore,  $q_{1,2,3} = 0.32q$ ,  $\Delta X_1 = 77$  mm,  $\Delta X_2 = 42$  mm, and  $\Delta X_3 = 74$  mm. These nonlinear size limits are determined simply by determining the limits of sizes for which  $1/n$  of the accommodated users ( $950/n$  in this example) in the virtual population are attributed to each size variant, where  $n = 3$ .

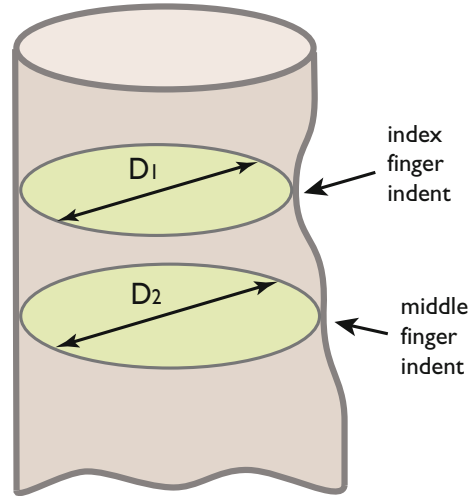
### 22.4.4 Example 2: Tool Handle

#### 22.4.4.1 Background

In the tool handle example, the objective is to specify the handgrip such that performance across the population of target users is high. Tool handles are commonly designed for the average user, making this an interesting example for the application of sizing. This is an additional opportunity for platforming within a power tool product family, where battery packs and motors are commonly standardized (Meyer and Lehnerd 1997). Proper tool sizing benefits the user by increasing comfort and safety. Poorly designed tools increase the risk of acute trauma and chronic disorders, such as carpal tunnel syndrome (Mital and Kilborn 1992), so ensuring the use of well-designed tools at home and in the workplace is important. The cross-sectional size and shape of a tool handle are the parameters of greatest interest in its design and have been the focus of previous work on this subject, which has been relatively well-studied for decades (Chengular et al. 2004; Mital and Pennathur 1999; Chaffin et al. 1999). In general, circular or elliptical handle cross-sections are recommended.

Some studies have attempted to generally correlate optimal handle size with user anthropometry, particularly *grip diameter* (also called *grip breadth*). Grip diameter is the diameter of the largest cylindrical object that can be grasped with the tip of the middle finger touching the tip of the thumb. Grant et al. (1992) found that, of three handle options (one with a 1 cm finger overlap, one with fingers just touching, and another with a 1 cm gap), grip strength was maximized with the smaller diameter

**Fig. 22.9** The handle assumes a circular cross section, and relevant parameters are the diameter of the first and second finger indents



handle in which the fingers overlap (Fig. 22.9). This configuration requires that the circumference of the optimal handle be equivalent to the user's grip circumference (i.e.,  $D_{grip} * \pi$ ) minus the constant  $C$ , where the optimal case is represented by  $C = 10$  mm. Handle diameter,  $D_{opt}$ , is then determined by dividing this handle circumference by  $\pi$ . Eq. (22.12) expresses this relationship mathematically.

$$D_{opt} = (D_{grip} * \pi - C) / \pi \quad (22.12)$$

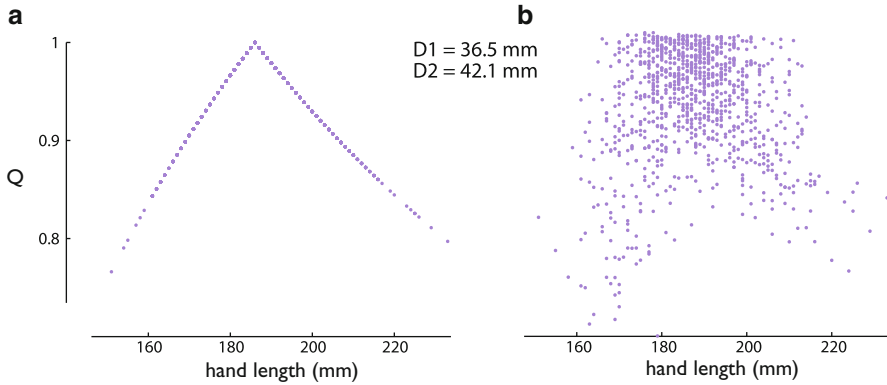
It has been shown that the index and middle fingers contribute most significantly to the gripping force of the hand (Kong and Lowe 2005). Therefore, the posture of the index and middle fingers is considered in a task that requires optimal grip strength for tool stability and security. A tool handle is considered that has variable cross-sectional properties to best accommodate the differing grip diameters of the index and middle fingers. Thus, the optimal shape of a cylindrical tool handle, such as the one shown in Fig. 22.9, with respect to its dimensions in the first and second finger indent, is desired.

For this work, the optimal diameter is determined by Eq. (22.12); however, this equation must be expanded to consider optimal handle diameter for both the first and second fingers. Equation (22.13) gives modified optimal diameter, where the indices 1, 2 correspond to index and middle fingers, respectively.

$$D_{opt}|_{1,2} = (D_{grip}|_{1,2} * \pi - C) / \pi \quad (22.13)$$

$D_{grip}$  indicates grip diameter (for index and middle fingers), and  $C$  represents the constant by which the fingers overlap for optimal strength, which is taken to be 10 mm for this study. The target user population was identified as ANSUR with equal representation from men and women. Regression was performed using hand





**Fig. 22.10** Grip quality plotted for each member of the population as a function of hand length for one size, without and with the residual variance component of the regression equation for grip diameter, (a) and (b) respectively. Optimal handle diameter for the first and second finger indents is also shown

data from Research Institute of Human Engineering for Quality Life (1997) to relate  $L_H$ —measure available in ANSUR—to  $D_{grip}$ . The residual variance was retained as described earlier, producing the following equations for the two grip diameters:

$$D_{grip|2} = 0.244L_H + N(0, 2.579 \text{ mm}) \quad (22.14)$$

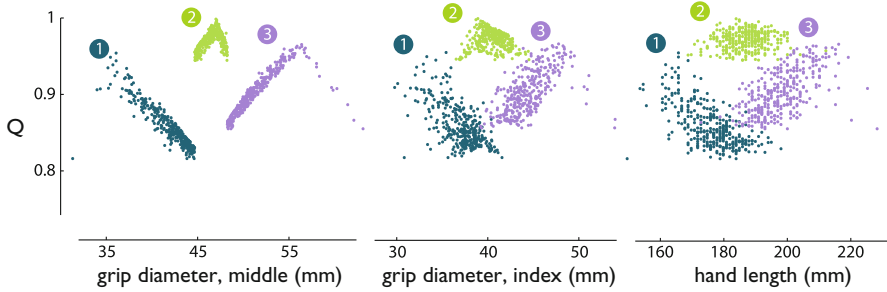
$$D_{grip|1} = 0.877D_{grip|2} + N(0, 1.180 \text{ mm}) \quad (22.15)$$

$N(0, RMSE)$  simply indicates a normal distribution with mean 0 and standard deviation equal to the root-mean-square error (RMSE) of the regression. The relevant measure from ANSUR,  $L_H$ , was randomly sampled from 500 males and 500 females from the ANSUR data. These 1,000 virtual users are the subjects for the virtual fitting trial performed to calculate grip quality,  $Q$ , for the population of users. The full details of this work are available in Garneau and Parkinson (2012).

#### 22.4.4.2 Single Size

Figure 22.10b illustrates grip quality as a function of hand length. Figure 22.10a simply uses a proportionality constant to compute  $D_{grip|1}$  and  $D_{grip|2}$  in Eq. (22.15). This proportionality constant is given by Eq. (22.15) without the part representing the residual variance. Figure 22.10b uses the entire relationship as given by Eq. (22.15) to compute the two grip diameters. Including the residual variance leads to greater scatter in the plots. All subsequent figures will use the relationship for grip diameter including residual variance.

While many products are offered in a single size, any platforming or modularity would have to exist across demographics (e.g., consumer and professional) or



**Fig. 22.11** Grip quality as a function of anthropometry for the case in which minimum grip quality is to be maximized across three handle sizes, from smallest to largest: blue ①, green ②, and purple ③. Optimal handle diameter for the first and second finger indents is also shown

related product lines. This simple, single-size example is included here for two reasons: (1) to demonstrate the approach and (2) to establish the necessary background for the multi-size example below.

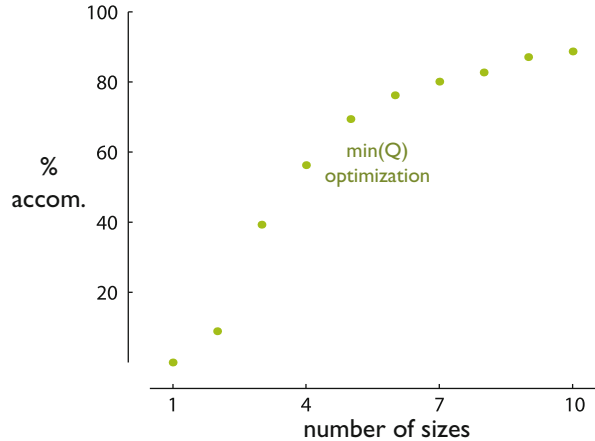
### 22.4.5 Multiple Sizes

Designs are evaluated by the minimum grip quality they provide (i.e., their performance is based on the worst experience of a user). The objective in increasing the number of handle sizes is to improve this minimum fit,  $Q$ . Figure 22.11 shows grip quality as a function of index finger grip diameter, middle finger grip diameter, and hand length, respectively, for the case in which minimum grip quality for each size is to be maximized.

Figure 22.11 shows that grip quality peaks for users with hand dimensions yielding optimal diameters ( $Q = 1$ ). The number of peaks in the figures indicates the number of optimal diameters (i.e., the number of sizes). In each case, an equal number of users are attributed to each handle size. The relationship between grip quality and hand length is nonlinear (i.e., possesses scatter) because the optimization is performed over two anthropometric variables for each person. Since the ratio between these variables is not constant, there is some deviation in grip quality for any given measure of the hand.

Consider a scenario in which only users with a grip quality,  $Q$ , of some minimum value are considered to be accommodated. Figure 22.12 plots the percentage of the 1,000 virtual users that achieve a grip quality  $Q \geq 0.95$  for 1–10 sizes, for both optimization schemata.  $Q \geq 0.95$  is an arbitrary goal; increasing or decreasing the required value of  $Q$  will affect accommodation levels. Studies of safety/comfort or a cost–benefit analysis would guide designers in choosing an appropriate minimum value of  $Q$  for accommodation.

**Fig. 22.12** The percentage of accommodated users is illustrated for a varying number of handle diameters for both optimization schemata. Accommodation is achieved for users with a grip quality of 0.95 or greater



## 22.5 Opportunities for Platforming and Modularity

Product platforming and modularity can be effective in most scenarios involving sizing or the allocation of adjustability. There are, however, several situations where DfHV offers opportunities that are particularly promising or that might be overlooked. The following is a description of these scenarios and an explanation of why platforming and modularity might be a good approach. Obviously this is not a comprehensive list, but these are situations that occur often and where platforming has not been considered.

### 22.5.1 Long-Tailed and Skewed Distributions

Most measures of length (e.g., stature, leg length) and many outcomes (e.g., uncensored, preferred fore-aft seat position in a vehicle) are approximately normally distributed. Consequently, designs that target the average body size or outcome measure are efficient in their ability to accommodate a large number of users. This is due both to the nature of the distribution and the adaptability of users. Consider Fig. 22.13, a probability density plot showing the allocation of individuals across a normally distributed measure like stature. The population (ANSUR men) exhibits a range of 545 mm across the measure, but 50 % of the members of the population are within 46 mm of the mean. Increasing accommodation from 50 to 90 % means the design must address nearly 2.5 times more variability.

When accommodation targets are high (e.g., 95 %), the amount of variability that the design must address can also be very high.

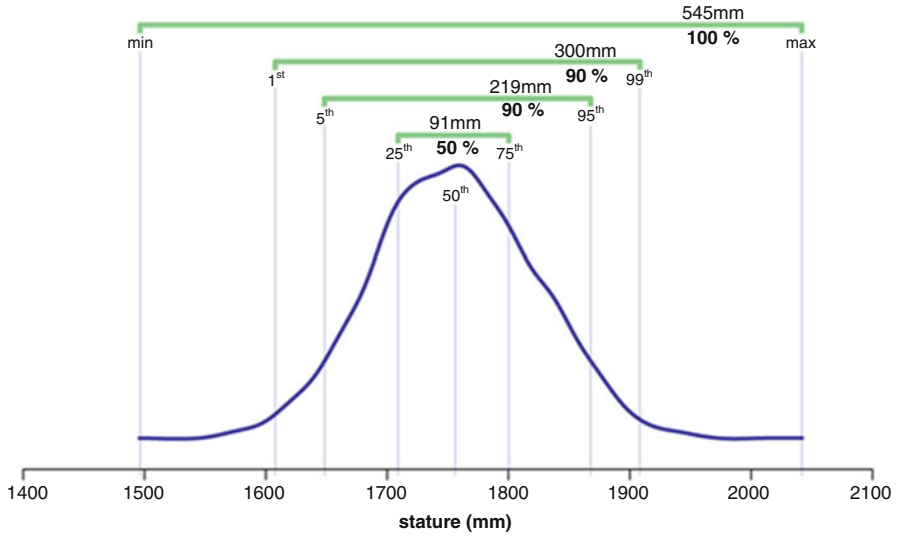


Fig. 22.13 A probability density plot for the distribution of stature for the ANSUR population

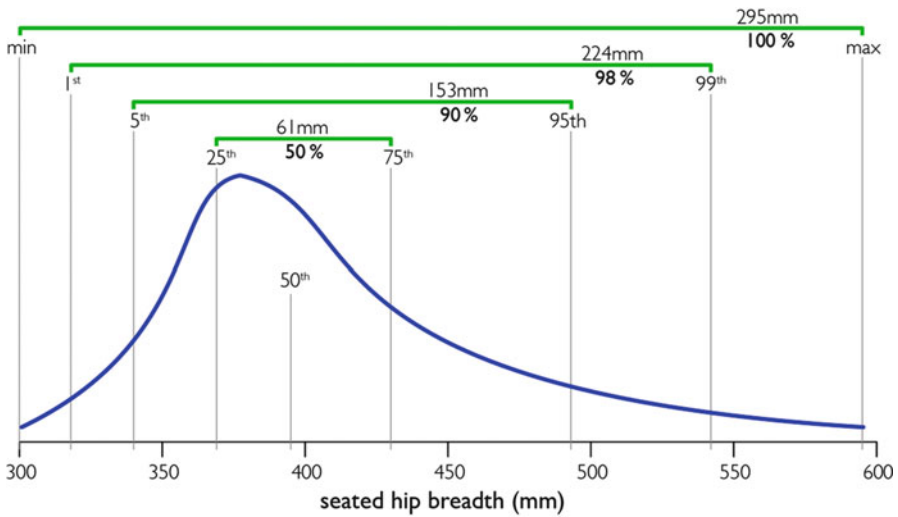


Fig. 22.14 A probability density plot showing the distribution of seated hip breadth for the US civilian population

Not all measures are normally distributed. In particular, many measures of width are known to be skewed, with a long right tail. Consider Fig. 22.14, which shows the probability density of seated hip breadth for the combined population of men and women in the US. As with a normal distribution, the 50 % in the most densely

distributed region of the population exhibit only a fraction of the large amount of total variability. There are, however, two important attributes of populations with skewed distributions that make them particularly amenable to platforming. First, the truncated left tail means that variability is limited in one direction, somewhat simplifying the problem when high levels of accommodation are desired. Second, the one-sided tail is often an indication that two platform variants might be appropriate: one for the distribution corpus and one for the tail.

When the distribution of anthropometric measures is one-sided or skewed, the range of design requirements might necessitate two or more variants.

### ***22.5.2 Segmented Populations and Disproportionate Disaccommodation***

Segmentation in users provides a number of opportunities for platforming. For example, the ratio of women to men in the workforce is known to vary across occupations (Jarman et al. 2012). Designers targeting a given segment should look at the distribution of variability in relevant user metrics such as body size, shape, capability, or preference and determine if any clustering occurs. Failure to do this can result in situations where particular demographic groups are disproportionately disaccommodated. In addition to reducing market opportunities, this can increase risk exposure for some groups, increasing their risk exposure (Elders et al. 2004; Ahonen and Benavides 2006; Buchanan et al. 2010). Consider, for example, a manufacturing operation that requires workers to be standing while performing it. An accommodation level of 95 % seems appropriate for this task—19 out of every 20 individuals could perform the task safely. However, if the work environment were designed such that the tallest 95 % of the US Civilian population were accommodated, 99 % of the disaccommodated individuals would be women (US Centers for Disease Control and Prevention 2010). Of the three major racial/ethnic groups in the USA, Hispanics tend to be the shortest. In fact, one in four Hispanic women would not be able to perform the task safely. In contrast, nearly 100 % of all men, regardless of race/ethnicity, would be accommodated.

In these situations of disproportionate disaccommodation, designers are in a difficult situation. Do they design an artifact, or environment based on the known demographics of the users? For example, ~90 % of truck drivers are men, so should truck cabs be designed to the variability exhibited by men? Alternately, the limited presence of women in that work force could be at least partially due to disaccommodation in the truck cabs. It is possible that participation of women would improve if the cabs accommodated a broader range of users. Product platforming and modularity are excellent approaches to addressing these needs.

Population attributes such as gender (Mohammed 2005), race/ethnicity (Hawes et al. 1994), and age (Annis 1996) have a strong influence on the distribution of anthropometry in a population. Due to these differences, a product may perform differently across race and gender groups. When designing for human variability, it is important to identify the user population and select representative anthropometric distributions that accurately estimate variability within it (Nadadur and Parkinson 2010). For example, an analysis of data from a US military population, ANSUR, found that 36 % of the 100+ anthropometric measurements showed significant differences between racial and gender groups (Walker 1993).

Products designed for international markets must take into account the anthropometric differences between various populations. Facemasks designed for an American population might disaccommodate up to 35 % of Chinese users (Yang et al. 2007). Even within neighboring groups, populations can have statistically different anthropometry; a comparison between Chinese, Japanese, Korean, and Taiwanese men found that not only are the individual measures different, the proportions between measurements are as well (Lin et al. 2004). On the other hand, two completely separate populations may be similar, such as that of Mexican female workers and Indian female agricultural workers (Dewangan et al. 2008).

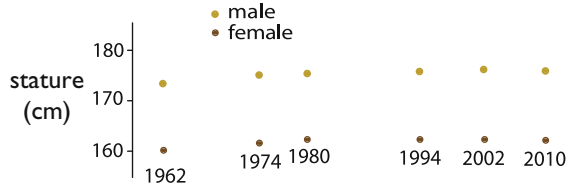
Identify different segments of users within the population and collect, synthesize, or otherwise estimate variability in size, shape, capability, and preference across these segments. Analyze this variability for sizing- and adjustability-related platforming/modularization opportunities.

### ***22.5.3 Long-Lifetime Products and Secular Trends***

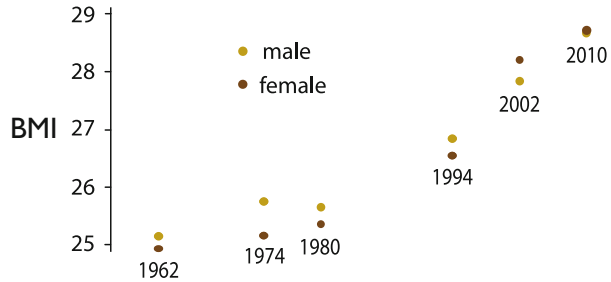
Many portfolios consist of products that have long usage lifetimes; this is more likely for products that are relatively expensive. Examples include heavy trucks, airplanes, and manufacturing operations. Changes in the target user population over the product's lifetime can mean that the needs of users as the product nears the end of its life can be very different than those at the beginning. There are three types of changes to which designers must be sensitive: secular trends, preference, and demography.

A secular trend is a continuous, noncyclical, increase or decrease in some measure over time. For example, as nutrition and health care have improved over the last century, there has been a significant increase in life expectancy. This, along with other factors such as birthrate, has resulted in a situation where the average age of populations in many countries is increasing (Department of Economic and Social Affairs—Population Division 2002). This “aging population” creates a number of

**Fig. 22.15** The average stature for men and women in the USA from 1962 to 2010



**Fig. 22.16** The average body mass index (BMI), a measure of weight for stature, for men and women in the USA from 1962 to 2010



opportunities for designers. There are other important secular trends related to DfHV. For example, across the world, the average height is increasing within a given racial/ethnic group. For countries with relatively homogeneous populations such as Scandinavia, this has resulted in dramatic increases in average height for the region. In areas where immigration is high, the average height might remain the same, but the standard deviation increases. In the US, for example, Whites, Blacks, and Hispanics (the three major racial/ethnic groups) are all getting taller, but the percentage of Hispanics in the US is increasing. Since that group tends to be somewhat shorter than the others, the net effect has been to stabilize the mean (Fig. 22.15), while the population as a whole gets both shorter and taller.

Another important secular trend is the increased prevalence of obesity. This is a global problem, but due to the ongoing survey conducted by the US Centers for Disease Control, it is best documented in the US. Figure 22.16 shows the dramatic increase in BMI over the last 30 years. With this change have come commensurate changes in body size and shape. For measures of width, this has both increased the mean and increased the length of the right tail of the distribution for most measures of width and depth.

Demographic changes can also cause secular trends in user groups. For example, increased numbers of women in the workforce, older workers, and more diverse populations can change the spatial requirements and physical abilities of user groups. The population for which a truck cab was designed in 1980 is substantially different than the population that is using it now. While it is impossible to precisely predict these changes, successful product portfolios will be adaptable to them. Platforming and modularity must consider these changes and look to achieve the requisite levels of flexibility. Although the magnitude of the change cannot be

anticipated, a designer might determine which aspects of the user population can change and how they might affect a given design. Platform variants can be identified for present and future times. Then, at appropriate points in the product's lifecycle, the platform can be leveraged to update the product to the needs of the new population (Nadadur et al. 2011).

Product portfolios that will be in use for a long period of time will likely witness the effects of secular trends in their users. Plan accordingly.

## 22.6 Conclusion

Product platforming is a powerful strategy. When used in combination with DfHV practices, it can improve the ability of a designer to meet the needs of a wider range of target users. The two case studies provide an introduction to current practice in sizing and the allocation of adjustability. Within the context of these two strategies, several specific opportunities for the application of DfHV within product platforming have been identified. These include products intended for global or diverse populations, those for which high levels of accommodation are expected, long-lifetime products, the opportunity to mitigate disproportionate disaccommodation, and designs that typically achieve poor performance because of the high or skewed variability in the attributes of target users. Product platform design has a rich history in mechanical and electrical components. Although not part of the typical product platform community, the practice of platforming around body size and shape, capability, and preference is an ancient one. Including these as potential factors in platforming opens many new opportunities for expanding platforming and improving efficiencies.

**Acknowledgments** Special thanks to Charlotte de Vries, Eliza Detweiler, and the research assistants and alumni of the OPEN Design Lab at Penn State University.

## References

- Ahonen EQ, Benavides FG (2006) Risk of fatal and non-fatal occupational injury in foreign workers in Spain. *J Epidemiol Community Health* 60(5):424–426
- Annis JF (1996) Aging effects on anthropometric dimensions important to workplace design. *Int J Ind Ergon* 18:381–388
- Bittner AC (2000) A-CADRE: advanced family of manikins for workstation design. In: *Proceedings of the human factors and ergonomics society, Long Beach, CA*, pp 774–777
- Buchanan S, Vossen P, Krause N, Moriarty J, Frumin E, Shimek JAM, Mirer F, Orris P, Punnett L (2010) Occupational injury disparities in the us hotel industry. *Am J Ind Med* 53(2):116–125



- Centers for Disease Control and Prevention (2010) National health and nutrition examination survey 2007-2010. National Center for Health Statistics, Hyattsville, MD
- Chaffin DB, Andersson GBJ, Martin BJ (1999) Occupational biomechanics, 3rd edn. Wiley, New York, NY
- Chengular SN, Rodgers SH, Bernard TE (2004) Kodak's ergonomic design for people at work, 2nd edn. Wiley, New York, NY
- Clauser C, McConville J, Young J (1969) Weight, volume and center of mass of segments of the human body. Technical Report AMRL-TR-69-70, Wright Patterson Air Force Base, Dayton, OH
- Colombo G, Cugini U (2005) Virtual humans and prototypes to evaluate ergonomics and safety. *J Eng Des* 16(2):195–203
- Department of Economic and Social Affairs—Population Division (2002) World population ageing: 1950-2050. UN DESA, New York, NY
- Dewangan K, Owary C, Datta R (2008) Anthropometric data of female farm workers from north eastern India and design of hand tools of the hilly region. *Int J Ind Ergon* 38(1):90–100
- Diffrient N, Tilley A, Bardagjy J (1981) Humanscale. MIT Press, Cambridge, MA
- Drillis R, Contini R (1966) Body segment parameters. Office of Vocational Rehabilitation Engineering and Science, New York, NY, cited in Clauser et al. (1969)
- Elders L, Burdorf A, Ory F (2004) Ethnic differences in disability risk between Dutch and Turkish scaffolders. *J Occup Health* 46:391–397
- Flannagan C, Manary M, Schneider L, Reed M (1998) Improved seating accommodation model with application to different user populations. In: Proceedings of the SAE international congress and exposition, vol 1358. SAE, Warrendale, PA, USA, pp 43–50
- Fromuth R, Parkinson M (2008) Predicting 5th and 95th percentile anthropometric segment lengths from population stature. In: Proceedings of the ASME international design engineering technical conferences, New York, NY, DETC2008-50091
- Garneau C, Parkinson M (2007) Including preference in anthropometry-driven models for design. In: ASME design engineering technical conferences, ASME International, Las Vegas, NV
- Garneau C, Parkinson M (2011) A comparison of methodologies for designing for human variability. *J Eng Des* 22(7):505–521
- Garneau C, Parkinson M (2012) Optimization of product dimensions for discrete sizing applied to a tool handle. *Int J Ind Ergon* 42(1):56–64
- Gite L, Majumdar J, Mehta C, Khadatkar A (2009) Anthropometric and strength data of Indian agricultural workers for farm equipment design. All India Coordinated Research Project on Ergonomics and Safety in Agriculture, Central Institute of Agricultural Engineering, Bhopal, MP
- Gordon CC, Churchill T, Clauser CE, Bradtmiller B, McConville JT, Tebbetts I, Walker RA (1989) 1988 anthropometric survey of US Army personnel: methods and summary statistics, Final report. Technical Report NATICK/TR-89/027, US Army Natick Research, Development and Engineering Center, Natick, MA
- Grant KA, Habes DJ, Steward LL (1992) An analysis of handle designs for reducing manual effort: the influence of grip diameter. *Int J Ind Ergon* 10:199–206
- Haslegrave CM (1986) Characterizing the anthropometric extremes of the population. *Ergonomics* 29(2):281–301
- Hawes MR, Sovak D, Miyashita M, Kang SJ, Yoshihuku Y, Tanaka S (1994) Ethnic differences in forefoot shape and the determination of shoe comfort. *Ergonomics* 37(1):187–196
- HFES 300 Committee (2004) Guidelines for using anthropometric data in product design. Human Factors and Ergonomics Society, Santa Monica, CA
- Information Center NHS (2009) Health survey for England. Department of Health, England
- Jarman J, Blackburn R, Racko G (2012) The dimensions of occupational gender segregation in industrial countries. *Sociology* 46(6):1003–1019
- Kong YK, Lowe BD (2005) Optimal cylindrical handle diameter for grip force tasks. *Int J Ind Ergon* 35(6):495–507

- Lehnerd A (1987) Revitalizing the manufacture and design of mature global products. In: Guile B, Brooks H (eds) *Technology and global industry: companies and nations in the world economy*. National Academy Press, Washington, DC, pp 49–64
- Lin YC, Wang MJJ, Wang EM (2004) The comparisons of anthropometric characteristics among four peoples in East Asia. *Appl Ergon* 35(2):173–178
- McCulloch C, Paal B, Ashdown S (1998) An optimisation approach to apparel sizing. *J Oper Res Soc* 49(5):492–499
- Meyer M, Lehnerd A (1997) *The power of product platforms*. Free Press, New York, NY
- Michalek J, Ceryan O, Papalambros PY, Koren Y (2006) Balancing marketing and manufacturing objectives in product line design. *ASME J Mech Des* 128(6):1196–1204
- Mital A, Kilborn A (1992) Design, selection and use of hand tools to alleviate trauma of the upper extremities: part ii—the scientific basis (knowledge base) for the guide. *Int J Ind Ergon* 10:7–21
- Mital A, Pennathur A (1999) Chapter 8: Hand tools. In: Kumar S (ed) *Biomechanics in ergonomics*. Taylor & Francis, London
- Mohammed YAA (2005) Anthropometric characteristics of the hand based on laterality and sex among Jordanians. *Int J Ind Ergon* 35(8):747–754
- Nadadur G, Parkinson M (2009) Using designing for human variability to optimize aircraft seat layout. *SAE Int J Passeng Cars Mech Syst* 2(1):1641–1648
- Nadadur G, Parkinson M (2010) Consideration of demographics and variance in regression approaches to estimating body dimensions for spatial analysis of design. *ASME J Mech Des* 132(2)
- Nadadur G, Garneau C, de Vries C, Parkinson M (2011) A real options-based approach to designing for changing user population of long-lifetime products. In: *Proceedings of the ASME international design engineering technical conferences*. ASME International, Washington, DC, DETC2011-48712
- Osteras T, Murthy DNP, Rausand M (2006) Product performance and specification in new product development. *J Eng Des* 17(2):177–192
- Parkinson M, Reed M (2006a) Improved head restraint design for safety and compliance. In: *Proceedings of the ASME international design engineering technical conferences*, ASME International, Philadelphia, PA, DETC2006-99429
- Parkinson M, Reed M (2006b) Optimizing vehicle occupant packaging. Technical report SAE Technical Paper No. 2006-01-0961, SAE International, Warrendale, PA
- Parkinson M, Reed M, Kokkolaras M, Papalambros P (2007) Optimizing truck cab layout for driver accommodation. *ASME J Mech Des* 129(11):1110–1117
- Reed M, Flannagan C (2000) Anthropometric and postural variability: limitations of the boundary manikin approach. Technical Paper 2000-01-2172. *SAE Trans J Passeng Cars Mech Syst* 109
- Reed MP, Manary MA, Flannagan CAC, Schneider LW (2000) The effects of vehicle interior geometry and anthropometric variables on automobile driving posture. *Hum Factors* 42(4):541–552
- Research Institute of Human Engineering for Quality Life (1997) Japanese body size 1992–1994. Osaka, Japan. <http://www.dh.aist.go.jp/bodyDB/s/s-01-e.html>
- Roe R (1993) Occupant packaging. In: Peacock B, Karwowski W (eds) *Automotive ergonomics*. Taylor & Francis, London, pp 11–42
- SAE International (2006) *Automotive engineering handbook*. SAE International, Warrendale, PA
- Statistisches Bundesamt Deutschland (2004) Mikrozensus. <http://www.destatis.de/DE/Meta/-AbisZ/Mikrozensus.html>
- Sundin A, Ortegen R (2006) Chapter 39: Digital human modeling for CAE applications. In: Salvendy G (ed) *Handbook of human factors and ergonomics*, 3rd edn. Wiley, New Jersey, NJ
- The State Bureau of Technical Supervision (1989) Human dimensions of Chinese adults (GB 10000-88)
- Tryfos P (1985) On the optimal choice of sizes. *Oper Res* 33(3):678–684

- Tryfos P (1986) An integer programming approach to the apparel sizing problem. *J Oper Res Soc* 37(10):1001–1006
- UGS (2007) Tecnomatix Jack UGS
- Van der Vegte W, Horvath I (2006) Including human behavior in product simulations for the investigation of use processes in conceptual design: a survey. In: Proceedings of the ASME international design engineering technical conferences, ASME International, Philadelphia, PA, DETC2006-99541
- Walker RA (1993) The impact of racial variation on human engineering design criteria. *NAPA Bull* 13(1):7–21. doi:[10.1525/napa.1993.13.1.7](https://doi.org/10.1525/napa.1993.13.1.7)
- Yang L, Shen H, Wu G (2007) Racial differences in respirator fit testing: a pilot study of whether american fit panels are representative of chinese faces. *Ann Occup Hyg* 51(4):415–421
- You H, Ryu T (2005) Development of a hierarchical estimation method for anthropometric variables. *Int J Ind Ergon* 35(4):331–343
- Zou T, Mahadevan S (2006) Versatile formulation for multiobjective reliability-based design optimization. *ASME J Mech Des* 128(6):1217–1226

**Part IV**  
**Applications and Case Studies**

# Chapter 23

## Building, Supplying, and Designing Product Families

David M. Anderson

**Abstract** Product families must be based on customer/marketing feasibility, operational flexibility, supply chain responsiveness, and design versatility. This chapter discusses the general strategies of applying these criteria to the fulfillment of product families.

### 23.1 Structuring and Selling Product Families

Product families must be based on all the following criteria: customer/marketing feasibility, operational flexibility, supply chain responsiveness, and design versatility. Product families are probably not going to be based on products that appear on adjacent pages in catalogs or web sites. Popular views about product families may concentrate on offering whatever customers want or whatever engineering can design. However, the supply chain and operations must be able to (a) provide readily available parts and materials for each product family and (b) build unique products in the family quickly without setup delays and costs.

To accomplish this, families of products should be structured to satisfy all the above criteria. Product families can be defined by starting with a broad range of product variations that would satisfy market needs and be wanted by customers—not just your current customers but also other potential customers—as discussed in many of the previous chapters. From that pool, the grouping would be explored as follows.

First, ascertain how much variety could be built in *existing* manufacturing facilities with *existing* parts, subassemblies, and materials (hereafter all called “parts”) and be done quick enough and inexpensive enough to satisfy the identified markets. However, unless the company has been implementing lean production,

---

D.M. Anderson (✉)  
Build-to-Order Consulting, Cambria, CA, USA  
e-mail: [anderson@build-to-order-consulting.com](mailto:anderson@build-to-order-consulting.com)

standardization, and other flexible manufacturing initiatives, current operations will probably not be adequate for a viable product family strategy.

If this is the case, then the next step would be to try to improve the flexibility of the current equipment with flexible fixtures and tooling, setup reduction efforts, quick-change tool techniques, rapidly downloaded computer numerically controlled (CNC) machine tool programs, dock-to-line part distribution, improved flow of parts and products, and efforts to resolve bottlenecks and fill in “missing link” processing equipment.

If it looks like this would still not be adequate for a viable product family strategy, then prospective families would have to be *designed* into coherent product families that can be built quickly at low cost from readily available parts and materials. The following sections discuss how to build product families flexibly, resupply parts and materials to all points of use, and design product families for flexible operations.

### ***23.1.1 What About Products That Do Not Fit into Families?***

Do not attempt to build nonfamily products in flexible operations where they do not fit.

Flexible operations offer enormous benefits to build a wide range of product variations in each predefined family at low cost with fast response. However, those benefits are dependent on (a) flexible manufacturing lines that were created for predefined variety (as discussed in the next major section) and (b) standard parts that are always available at all points of use (as presented in the following section). Incompatible products include variations outside predetermined families, legacy products, spare parts, prototypes, and acquired products.

Here are the options for product variations that do not fit into any product family—the orphans—and are not compatible with flexible operations and supply chains. The first is to expand the family to include valuable variations, with appropriate enhancements in operations and supply chain management.

The second is to perform product line rationalization (Anderson 2008, Chap. 3) to identify products that are incompatible or not really reaching goals for profitability, which is only meaningful if all costs are quantified (Anderson 2008, Chap. 12; see Sect. 23.8.2 at the end). This procedure may recommend dropping such products, redesigning them to fit into flexible operations, outsourcing them, selling their rights, or moving their production to a separate facility or isolate their production into a self-supporting profit-and-loss center whose staffing and overhead are totally outside flexible operations. To be self-sustaining, this operation may have to raise prices to pay for all its cost. If the market will not “bear” these costs, then those products will have to be improved or rationalized away.

### **23.1.2 *Focusing on the Families***

Sales and marketing must confine sales to the designated product families. If there are marketing opportunities outside these families, the options are (a) marketing works with engineering, manufacturing, and supply chain management to expand existing families or create new ones; and (b) until this happens, the closest family version could be modified by skilled people in the independent profit-and-loss center mentioned above. On a total cost basis, this will cost more than flexible operations and will take longer to build, but it may keep certain market opportunities alive until the variations can be incorporated into flexible operations.

## **23.2 Building Families of Product Quickly at Low Cost**

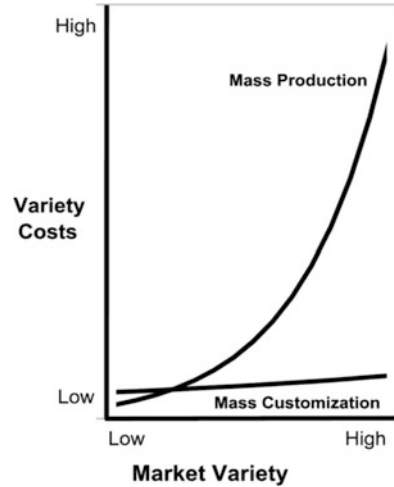
To maximize the sales of product families, all variations must be built cost-effectively and quickly, ideally on-demand as a build-to-order operation (Anderson 2008, Chap. 8) from always available parts (Anderson 2008, Chap. 7) without forecasts or inventory.

### **23.2.1 *Why Mass Production Methodologies Will Not Work for Product Families***

Mass production is very efficient for building identical products for the mass markets (Anderson 2008, Chap. 2). The epitome of mass production efficiency was the Ford Rouge River Model T plant, which could build any-color-as-long-as-it-is-black cars for \$245 in 1923. However, the cost of variety rises substantially with variety, as shown in Fig. 23.1 because of setup costs and inventory carrying costs, which presents one of manufacturing's most irreconcilable dilemmas: *raising batch size lowers the setup cost charge to each product but raises inventory and vice versa*. Material overhead also rises with variety to cover many parts that have to be ordered based on forecasts.

Forecasts also get substantially worse with variety, thus presenting manufacturing's other irreconcilable dilemma: *trading-off inventory vs. order response time*. This can either be good if the needed variation happens to be in stock but bad enough to thwart a product family strategy if customers have to wait for the next batch or if scheduled production is interrupted with expensive emergency runs with very high setup charges (the opening scene in Eli Goldratt's classic industrial novel, *The Goal* (Goldratt 1992)). Since product families would be at the high end of the variety curve, more flexible operations will be necessary for product families.

**Fig. 23.1** Variety costs as a function of market variety



### ***23.2.2 The Value of Building Low-Cost Family Variation Quickly***

Building and shipping product family variations quickly bring many benefits:

- Capturing more orders when the purchasing decision is influenced by fast delivery
- Avoiding delivery delays from inventory outages
- Increasing customer satisfaction, loyalty, and reorders
- Getting revenue back early
- Building at low cost and offering fast delivery will be an unbeatable competitive advantage

### **23.3 How to Build Product Families Flexibly**

Using some of the following methodologies will enable a manufacturer to quickly build any product in a product family. Utilizing all of these methodologies will enable build them on-demand and shipping them the same day (Anderson 2008, Chap. 8).

Most companies build products in batches or lots, which are scheduled, based on forecasts, after all the ordered parts and materials arrive. Then, the finished products are placed in a warehouse until they are ordered.

The batch size is determined by the setup times (in which rising values drive the batch size up) and cost of “carrying” inventory (in which rising costs drive the batch size down)—another endless trade-off dilemma. Further details follow.



### ***23.3.1 Setup and Batch Elimination***

Setup reduction techniques (Shingo 1985) can shrink the batch size, which will help moderate the variety situation, but for high-variety situations, setup needs to be eliminated to achieve “batch size of one” flexibility, which would be ideal for large product families.

If successive products are to be unique and different, then there cannot be any significant setup delays to get parts, change dies and fixtures, download programs, find instructions, or any kind of manual measurements, adjusting settings, or positioning of parts or fixtures. For a plant to mass-customize or spontaneously build families of products to order, all production setup must be eliminated, not just the low-hanging fruit or “as much as you can.” Setup “elimination” is defined as reduced to the point where it is still feasible to operate efficiently in a batch-size-of-one mode.

Part setup can be eliminated by eliminating kitting and distributing all parts to all points of use. Positioning setup can be eliminated by versatile jigs and fixtures. Tooling setup can be eliminated with universal tooling, rapid die changes, or quick-change inserts. Machining setup can be eliminated with optimal utilization of programmable CNC machine tools. Programming setup can be eliminated by downloading machine tool programs on-demand or generating them on-the-fly. Process variable setup can be eliminated by standardizing on process variables. Manual setup to find and understand instructions can be eliminated by displaying instructions on monitors.

### ***23.3.2 Tooling Setup Elimination***

First, design the product/processes to eliminate the need for tooling changes for cutting tools, dies, molds, tool plates, and fixtures. Design tool plates, jigs, and fixtures to be versatile enough to accept all parts in the family without having to change tool plates, jigs, and fixtures. The parts or standardized raw material must be able to be quickly positioned in fixtures without any need for manual positioning or measuring. Parts may need to be designed with common fixturing geometries.

If that is not possible, develop ways to change tooling rapidly. Clever, universal die and mold mounting geometries can be developed to facilitate quick changeovers (Shingo 1996). Conveyors and carousels, which were first applied to moving parts and products, are now being applied to moving dies and fixtures quickly in and out of machine tools.

### ***23.3.3 CNC to Eliminate Machining Setup***

CNC machine tools are very versatile tools to eliminate setup since many operations can be done by multi-axis machine tools without having to reposition the workpiece. CNC machining centers can perform a wide range of operations

such as machining, drilling, tapping, and so forth. The more operations that can be done in one operation, the fewer times the workpiece needs to be moved and set up. In fact, a key principle to ensure design for manufacturability (DFM) (Anderson 2013) and quality is to make sure all critical dimensions are cut in the same machine tool in the same clamping.

### ***23.3.4 Flow Manufacturing***

If setup can be eliminated or reduced enough to eliminate the need to manufacture in batches, then parts, subassemblies, and products can flow one piece at a time. One-piece flow assures the throughput and flexibility needed for flexible operations. U-shaped lines can facilitate feedback, make more people available to help colleagues nearby, and enhance visual monitoring and correction (visual control.)

### ***23.3.5 Assuring Quality with One-Piece Flow***

One-piece flow has distinct advantages for assuring quality. First, flow manufacturing help to eliminate the possibility that recurring defects may be built into several batches before being caught at a downstream assembly or inspection step. Second, people working in flow manufacturing look for any visible deviation as each part is handed to “its customer” (the next station). Further, if the part does not fit or work in the next operation, then the feedback will be immediate, leading to quick rectification of the problem.

### ***23.3.6 Flexible Source Cells***

Flexible source cells can make a wide variety of parts on-demand from standard raw materials without setup delays. For example, a flexible sheet metal source cell can feed standard sheet metal from a coil and cut any rectangular shape on-demand and then feed it directly to programmable cutters or be a Kanban source for other cells. Such a cell could be located in or near the predominant user line, with single sheet conveyance to other CNC machines or Kanbans.

To feed a flexible source cell from a coil, the cell must standardize on one sheet metal thickness, material, and grade, which could be the “better” version of all products that are fed by the cell. Fortunately, coils of standard material can be bought with confidence; it will be used one way or another.

### 23.3.7 *Flexible Assembly*

Flexible assembly stations would assemble cell-built parts and subassemblies with other standard parts and fasteners that are always available at assembly stations. Assembly instructions would be presented for each product variation on monitors or a paperwork “traveler” that accompanies a major part, for instance, a frame, chassis, circuit board, or major off-the-shelf part.

## 23.4 Spontaneous Supply Chains

Spontaneous supply chains (Anderson 2008, Chap. 7) play a key in product family strategies to avoid the need to generate forecasts, count inventory on hand, generate purchase order inputs through MRP (material requirements planning) systems, place purchase orders, wait for parts to arrive, expedite those that are late, receive (and maybe inspect) materials, warehouse, group into kits for scheduled production, and distribute within the plant. Flexible operations can avoid all these costly and time-consuming steps with a spontaneous supply chain, which is able to pull in standard materials and parts on-demand. It is highly unlikely that this can be accomplished merely by asking existing supply chains to deliver all of your existing parts on-demand.

The concepts presented herein are labeled as the *resupply* of parts and materials as opposed to procurement or purchasing. This is to emphasize that most of supply chain management is a resupply function of parts and materials that have been procured before. Parts and raw materials could be automatically resupplied using the following techniques (in order of increasing variety).

### 23.4.1 *Steady Flow of Standard Parts*

The ultimate scenario for spontaneous resupply is to reduce the number of part or raw material types within each category to *one*, in which case steady flows can be arranged for each one. Ideally, there should only be one type of each material or part. Then, forecasting multiple types would be unnecessary and “ordering” would be as simple as matching the tonnage in to the tonnage out; in other words, the incoming flow of the standard raw materials would be equal to the monthly consumption of the plant. These will be used one way or another. Multiple types in each category would allow the same spontaneity if material changeover is quick and the ratio is constant or predictable.

### 23.4.2 *Linear Cutoff*

Raw material variety can be greatly reduced by cutting off linear materials on-demand at the points of use or the cutoff station could be a Kanban source for

parts resupplied automatically to other points of use. The cutoff machine could be fully programmable or a worker could position the material up against a programmable, or manual, stop. Linear materials include all forms of bar stock, extrusions, strips, tubing, hose, wire, rope, cable, chain, and so forth.

### ***23.4.3 Material Cut-to-Length/Shape***

Raw material can be cut to length or shape on-demand from the longest version or standard size by programmable CNC equipment, such as laser cutters and screw machines, by single-axis programmable cutoff machines, or from less-automated tools based on online instructions.

### ***23.4.4 Min/Max Resupply***

The “min/max” technique is an effective way to automatically resupply raw material like sheet metal, where material is consumed until the stack reaches the “min” level, usually marked on the rack or wall. This triggers a reorder of the material to bring it up to the “max” level.

### ***23.4.5 Breadtruck***

The easiest and “lowest hanging fruit” in material logistics is the breadtruck (sometimes called “free-stock”) delivery system for small, inexpensive parts, like fasteners. Instead of counting on forecasts to trigger an MRP system to generate purchase orders, all the “jelly bean” parts can be made available in bins at all the points of use. A local supplier is contracted to simply keep the bins full and bill the company monthly for what has been used, much like the way bread is resupplied by the breadtruck to a small market.

### ***23.4.6 Kanban***

Kanban is a versatile technique that enables automatic resupply of parts that can be made in batches or have not-quite-spontaneous delivery times. In Kanban resupply, parts with limited variety are made, maybe in batches, and resupplied automatically to replenish parts bins based on part consumption. This is one of the many pull systems used to “pull” parts into assembly operations. The resupply is automatic once the pull signal gets to the supplier.

There are many simple ways to do this without complex information systems such as MRP or ERP. The most common is the two-bin system, which has two rows of parts. Initial assembly starts with all bins full of parts. When the part bin nearest the worker is depleted, then the full bin behind moves forward. The empty part bin is then returned to its “source,” which could be the machine that made the part, a subassembly workstation that assembled the part, or a supplier. The source fills the bin and returns it to this assembly workstation behind its counterpart which is still dispensing parts.

The beauty of Kanban resupply is that the system ensures an uninterrupted supply of parts without forecasts or high-overhead-cost ordering procedures. The number of parts in a bin is based on the highest expected usage rate and the longest resupply time. The size of each bin is determined by the bin quantity and size of the parts. For large parts, some companies use two-truck Kanbans, in which parts are drawn from one truck trailer while the other trailer goes back to the supplier for more parts. Alternate systems include Kanban squares for larger parts and a two-card system where the cards travel (or are faxed or e-mailed) back to the source instead of the bins. Electronic equivalents can also be utilized. Thus, Kanban resupply avoids the uncertainty of forecasting, the cost of purchasing, and the cost and risk of inventory.

Kanban works best for semi-standard parts without too much variety, which would increase work-in-process (WIP) inventory and clutter assembly stations with too many part bins. Kanban parts can be made in mass-produced batches. Of course, the parts manufacturers may have to implement setup and batch size reduction to be able to economically make batches small enough for Kanban deliveries.

### ***23.4.7 Parts Made On-Demand In-House***

In order for flexible manufacturing to work, all parts must be available on-demand. If there are any key parts that are not suitable for Kanban and no supplier can build them and ship them quickly enough to your pull signal, then you might have to bring those operations in-house and build them in flexible manufacturing operations.

### ***23.4.8 Strategic Stockpiles***

Until any of the aforementioned techniques can be implemented, it may be necessary to have strategic stockpiles of certain parts. The assembler could use selective stockpiles to temporarily compensate for any aspect of the supply chain that cannot be pulled or for temporary availability problems on standard parts.

### ***23.4.9 Dock-to-Line Deliveries***

To be truly agile, incoming parts and materials must flow directly to all points of use, called dock-to-line, instead of going through the traditional maze, which includes the receiving department (where they are logged in), the incoming quality control (IQC) department (where they are inspected), the raw material warehouse (where they are inventoried), and the kitting department (where they are counted and grouped into a batch worth of parts).

## **23.5 Designing Product Families**

Existing product designs may not be suitable for product family operations. The product portfolio may have too many unrelated products that lack any synergy and, thus, too many different parts and processes. There may be a needless and crippling proliferation of parts and materials. The specified parts may be too hard to get quickly. The products and processes may have too many setups designed in. Quality may not be designed into the product/process which results in disruptions when failures loop back for correction. The product/process design may not make optimal use of CNC as most CNC equipment is used in a batch mode, not flexibly. So if the existing production operations and supply chains cannot build product families, then they will need to be designed to make the product family strategy viable.

### ***23.5.1 Developing Products for Families***

To be successful at designing products for a product family strategy, multifunctional product development teams must design products in synergistic product families, design around aggressively standardized parts and raw materials, make sure specified parts are quickly available, consolidate inflexible parts into very versatile standardized parts, assure quality by design with concurrently designed process controls, and concurrently engineer product families and flexible flow-based processes.

Further, product development teams need to eliminate setup by design by specifying readily available standard parts and tools, designing versatile fixtures at each workstation that eliminate setup to locate parts or change fixtures, and making sure part count does not exceed available tool capacity or space at each workstation.

Finally, products must be designed to optimize the use of available programmable CNC fabrication and assembly tools, without expensive and time-consuming setup delays.

### ***23.5.2 Designing for No Setup***

Product design has a profound effect on setup. Excess proliferation of parts complicates internal part distribution and may make it impossible to have all parts available at all points of use. Even a moderate excess of part types will cause setup delays to distribute, find, and load parts into manual or machine bins. A serious excess of part types may make it necessary to kit parts for every batch, which is a significant setup. Designers can eliminate fixturing setup by designing parts for versatile fixturing which, if not already in place, may have to be concurrently designed with the parts.

Designers can eliminate tool change setup by designing parts around common tools (cutting tools, bending mandrels, punches, etc.), ideally one tool that never has to be changed. If multiple tools are required, then designers must keep tool variety well within tool changing capacity for the whole family.

Design for manufacturability (DFM) principles can greatly simplify assembly (Anderson 2013). Designers need to work with manufacturing engineers to concurrently develop simple assembly procedures that can be understood in a few seconds either on a computer screen or on paper instructions that can be quickly located and understood.

### ***23.5.3 Designing for CNC***

Computer numerically controlled machine tools (CNC) offer vast opportunities to eliminate machining setup, as discussed in Sect. 23.3.3. CNC machine tools include metal cutting equipment (mills, lathes, etc.), laser cutters, punch presses, press brakes, printed circuit board assemblers, and basically any production machine controlled by a programmable computer. Designers need to understand enough about CNC operation to use the versatility of CNC to eliminate setup.

### ***23.5.4 Designing Around Standard Parts and Materials***

Standardization of parts and material variety reduction are the most important design contributions to the feasibility of spontaneous supply chains. *Aggressive* standardization can enable the easiest technique of spontaneous supply chains: steady flows of very standardized parts and materials. If there are too many different parts and material types, then steady flows cannot be arranged because of the variety and unpredictability of demand. The total cost value of standardization and its contribution to the product family business model should motivate engineers and materials organizations to implement such aggressive standardization.

### ***23.5.5 Designing Around Readily Available Parts/Materials***

A spontaneous supply chain depends on readily available parts and materials. Therefore, it is an important aspect of engineers' jobs to specify parts and materials that are readily available. Usually, design engineers choose parts based on functionality and maybe quality, but for product families, availability is equally important.

### ***23.5.6 Design for Manufacturability***

Design for manufacturability guidelines can support product family strategy by minimizing incoming variety of parts and materials, for instance, by combining right/left-hand parts, combining parts and functions into a single part, specifying prefinished material, and avoiding arbitrary decisions (Anderson 2013).

### ***23.5.7 Arbitrary Decisions***

Successful product development requires that designers proactively design products for a product family environment using the principles presented herein. This may be difficult or impossible if designers make arbitrary decisions that preclude implementation of these principles. All design considerations—functionality, cost, quality, and flexibility—must be taken into account early. If they are not, then designers will probably make many arbitrary decisions that will make it much harder to satisfy omitted design considerations later.

## **23.6 Ways to Build Variety**

There are three ways to easily build variety for a product family strategy: (1) modular, (2) adjustable, and (3) dimensional means (Anderson 2008, Chap. 9). The most obvious way, modularity, can create variety by assembling various combinations of modules. Adjustability is a reversible way to create variety, as mechanical or electrical adjustments. Dimensionality involves a permanent cutting-to-fit, mixing, or tailoring. The key to success is to optimize the combination of all these techniques, not just the obvious modularity. This optimization comes from a thorough understanding of these techniques, many of which are discussed in the earlier chapters of this book.

### ***23.6.1 Postponement***

Postponement is a mass customization technique that is applicable for certain products that can have their variety postponed until just before shipping.



The factory builds basic “vanilla” platforms and quickly adds “flavors” upon receipt of order. Postponement is most suitable for product architecture that has a major platform part can be built without variation and then customized by various adjustments, configurations, or bolt-on modules.

### **23.6.2 Extending Product Families with Mass Customization**

The phrase mass customization was first coined in the book by Stan Davis (Davis 1987). Inspired by Davis’ book, B. Joseph Pine II then wrote the book, *Mass Customization: The Next Frontier in Business Competition* (Pine 1993), which makes an eloquent case for the paradigm.

Mass customization is the ability to quickly and efficiently build-to-order customized products. It uses all the techniques presented so far for the build-to-order of standard products and extends that to custom products. These products can be customized for individual customers or niche markets, such as versions optimized for certain market segments, industries, regions, or countries.

Prerequisites for mass customization are acquiring the ability to: (1) quickly and efficiently assemble products on-demand, as discussed above, and (2) procure materials and parts spontaneously. So in addition to offering predetermined product family variations, mass-customized products could be offered, thus offering even more appeal to customers and business opportunities for the company.

Engineers must concurrently engineer combinations of standard parts and modules with mass-customized parts and modules. One way to accomplish this is to design in adjustability, configurability, and programmability to create new variations based on customer input rather than predefined values by:

1. Creating parametric CAD templates in which CAD drawings are based on floating dimensions that can accept custom inputs that would “stretch” or “shrink” various dimensions.
2. Then the customized CAD drawing would be used to generate customized CNC machine tool programs.
3. The CNC machine tools then make customized parts from “stock” consisting of standard raw materials.
4. The custom part would then be assembled into the product through standard interfaces.

### **23.7 Synergies of Mass Customization and Product Families**

There is a natural synergy between product families built-to-order and mass customization. They share the same flexible operations and spontaneous supply chain. Product families and mass customization operations are equally efficient and very complementary. Adding mass customization to a product family strategy can

push the combined volume over the threshold that may be necessary to justify these implementations. Thus, mass-customized product families would be more likely to be approved and succeed.

## 23.8 Essential Prerequisites

### 23.8.1 *Standardization*

A successful product family strategy depends on standardization for several reasons. First, flexible manufacturing, by definition, depends on all parts being always available at all points of use, which can only be done for very standard parts. Too many parts would make it too hard to (a) fit them all into workstations and (b) assure availability spontaneously. So, if that is the case, supply chain management would have to order all the parts based on forecasts, which is how parts are ordered in mass production, which is not flexible and, therefore, not conducive for product families.

Second, for this availability, at all points of use, every part would have to be:

- (a) Versatile enough to support all the variations in the family. If this versatility is not found off-the-shelf, parts may need to be designed and built to have that versatility. For instance, molded or cast parts would be designed with all the features molded or cast in for various variations in the family.
- (b) High enough quality and tight enough tolerances to satisfy the most demanding variation in the family. For example, if some variations need 1 % tolerance resistors, then that becomes standard tolerance for all resistors, even if other variations might only need 5 % parts. The author applied this standardization for entire printed circuit board factories at Intel's Systems Group because the money saved in material overhead exceeds any increased part cost.
- (c) High enough performance or capacity for all family variations. For instance, if the most demanding variation in the family needs a 100 W power supply, then all variations would get a 100 W power supply, rather than struggling to supply several unusual power supplies for several wattages under 100 W.
- (d) Readily available from the source so each standard part selected would be the most available one within a range of candidates that meet or exceed the above criteria.

Third, flexible manufacturing must have dependable deliveries from suppliers and be able to distribute parts to all points of use in flexible factories. This can only be done with very standard parts.

Fourth, for spontaneous product manufacture on-demand, parts must be resupplied spontaneously with the spontaneous resupply techniques discussed earlier. These techniques work best with standard parts, but may not be feasible at all if incoming part variety is too great.

Last, product families will benefit from the cost-effectiveness of standardization and the feasibility of product families may depend on the efficiencies that include:

- Better purchasing leverage and economy-of-scale savings
- Material overhead that can be 1/10 of nonstandard parts, which reflects savings in purchasing efforts and encourages engineers to specify standard parts, thus lowering total cost
- Much less inventory carrying costs for parts and materials
- Less expediting and fewer change orders to solve availability problems

To realize, and get credit for, these benefits, all costs must be quantified as discussed next.

### ***23.8.2 Total Cost Quantification***

For a product family strategy to proceed, quantifying all costs is essential because most of the cost benefits—and most of the justification—will come from reducing overhead costs, some of which will be dramatic. The main cost savings will be inventory carrying costs, which average about 25 % of the inventory value per year (Anderson 2008, Chap. 2). So for every \$4 million of inventory, the company will have to pay \$1 million per year. A good product family strategy will consolidate many previous inventory SKUs into a few versatile product family platforms, thus greatly reducing finished goods inventory. Further, if build-to-order principles are pursued, then product family variations could be built on-demand without any finished goods inventory at all. Moreover, other significant overhead costs that can be eliminated (or, at least, greatly reduced) are setup costs, WIP inventory (that is eliminated by one-piece flow), and material overhead, which can be as low as one tenth when procuring standard parts.

Furthermore, standardization depends on total cost to (a) quantify the benefits cited in the above section and (b) ensure engineers embrace standardization. Better versatility, tolerances, performance, and availability will probably raise the “cost” of a bill of material line, although it will result in a much greater total cost savings for a net lower cost. However, if the total cost savings are not quantified, then designers will resist standardization because it may appear to raise the most visible cost: parts and materials.

## **References**

- Anderson DM (2008) Build-to-order & mass customization. CIM, Cambria, CA  
Anderson DM (2013) [www.design4manufacturability.com](http://www.design4manufacturability.com), accessed 30 July 2013  
Davis S (1987) Chapter 5: Mass customization. In: Future perfect. Addison-Wesley, Reading, MA  
Goldratt EM (1992) The goal, second revised edition. North River, Great Barrington, MA

Pine JB (1993) Mass customization. Harvard Business School Press, Boston, MA

Shingo S (1985) A revolution in manufacturing, the SMED system. Productivity, Portland, OR

Shingo S (1996) A summary of SMED principles for shop floor personnel, Quick changeover for operators, the SMED system. Productivity, Portland, OR

# Chapter 24

## Modular Function Deployment Applied to a Cordless Handheld Vacuum

Fredrik Börjesson

**Abstract** Modular Function Deployment (Erixon, Modular function deployment – a method for product modularization. PhD thesis, The Royal Institute of Technology, Stockholm, 1998) is a structured method used to define modular product architectures through the integration of customer values, company strategy, and the product technology. A modular product architecture breaks down a product into modules that can be directed toward specific, strategic goals where the operations of the company are optimized within each module individually. A presentation of the theory of Modular Function Deployment will serve as the groundwork for a case study based on a cordless handheld vacuum.

### 24.1 Theory of Modular Function Deployment

In the early 1990s Professor Anders Arnström in the Dept. of Production Engineering at KTH Royal Institute of Technology (KTH) in Sweden started a program to promote cell-based automated manufacturing in Swedish industry (Modular Management 2012a). Research during the program identified that many products were not designed to be produced using this manufacturing approach (Lange 2012). Products that were appropriate for cell-based manufacturing were found to be modular in their physical shape. Consequently, the research team began to develop a product development procedure that could produce a modular product. The result was Modular Function Deployment.

---

Modular Function Deployment and MFD are registered trademarks of Modular Management, Inc.

F. Börjesson (✉)

Modular Management USA, Inc., Bloomington, MN 55425, USA

e-mail: [fredrik.borjesson@modularmanagement.com](mailto:fredrik.borjesson@modularmanagement.com)

MFD as a research focus has generated a broad range of articles and continues to be taught at KTH as a product development method. Another outcome of the research program was the establishment of the consulting company, Modular Management, which has specialized in the application of Modular Function Deployment. Since the inception of the company in 1995, a wide range of product development efforts have applied MFD including cars, trucks, washing machines, dishwashers, front entry doors, commercial air handling units, construction equipment, and many more (Dobberfuhr and Lange 2009).

Complexity cost reduction is one of the common objectives of a modularity program. Unique Part Number Count (PNC) is a leading key performance indicator used to measure the level of reduction. PNC reductions typically range from 30 % (Modular Management 2012b, c) to 58 % (Modular Management 2012d) and in some cases as high as 90 % (Modular Management 2012e). Developing a modular product architecture also allows the company to offer more end product variants. Application of MFD at an early phase in product development allows a cross functional team to collaborate and define target market segments, product features, performance levels, and launch plans.

As a matrix-based method, Modular Function Deployment manages data efficiently and used software to perform numerical or statistical analyses. Hierarchical clustering (Romesburg 2004) may be utilized in MFD for the purpose of generating modules (Stake 2000; Hölttä et al. 2003). Algorithms are used to perform the clustering and are convenient for large matrices. Manual cluster can be performed with small and simple matrices.

Quality Function Deployment (QFD) (Akao and Mizuno 1994), Design Property Matrix (DPM: Nilsson and Erixon 1998), and Module Indication Matrix (MIM: Erixon 1998) are the three primary matrices utilized by the method to capture the product and its strategy. These three matrices are interlinked and referred to as Product Management Map (PMM), illustrated in Fig. 24.1.

Additional matrices, shown in gray in the above figure, can be used to enhance the result of MFD. Customer Value Rating (CVR) is used when the modular product is intended to satisfy the needs of multiple market segments each with unique product preferences. The Interface Matrix documents interactions between modules and allows for further optimization of the architecture.

Modular Function Deployment consists of five steps, Fig. 24.2. These five steps are used to capture the voices of the critical stakeholders in the formulation of the product including the customer, the engineering team, and the strategic initiatives of the company. Step one focuses on the Voice of Customer by first defining target market segments. Market segments are differentiated by their needs and preferences. These needs are quantified as Customer Values and rated by importance to each segment. Customer Values are transformed into a set of Product Properties which describes in metric-based terms how the product will meet the needs of the customer. The impact of Product Properties on Customer Values is captured in a Quality Function Deployment (QFD).

Voice of Engineering is addressed in the second step of MFD by selecting technical solutions. A technical solution is a physical entity designed to embody

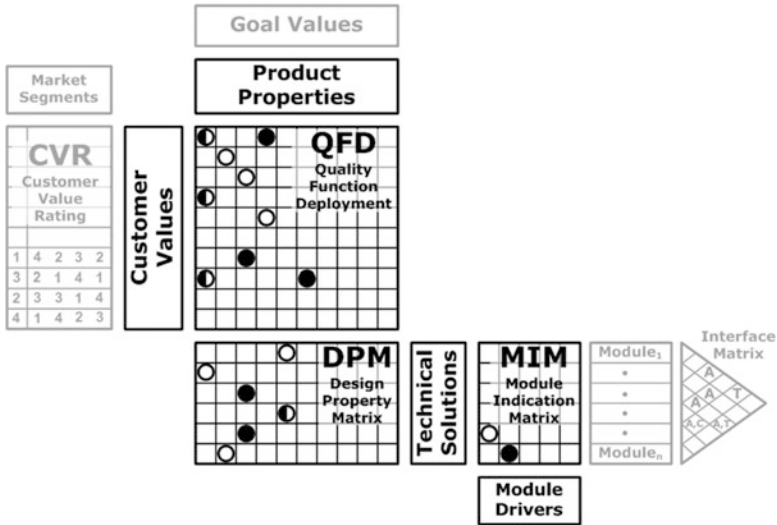


Fig. 24.1 PMM features three mandatory interlinked matrices

Product Properties and carry a required function in the product. To generate the list of technical solutions, the product is decomposed by utilizing either a top-down or bottom-up approach. Then the relationship between Product Properties and technical solutions is documented in a Design Property Matrix (DPM). It is possible that alternative technical solutions can be used to fulfill a function. In this case the solutions are evaluated using criteria from QFD, DPM, and internal sources to determine the appropriate direction for the product architecture.

A unique aspect of the method is the application of Voice of Business in the third step. Module drivers are the means by which a company’s strategic intent can be applied to the product structure. Module drivers are assigned to technical solutions in the Module Indication Matrix (MIM). Data from both DPM and MIM are statistically clustered to generate conceptual modules.

Module concepts are evaluated in the fourth step of MFD by defining the interfaces. An interface is a representation of an agreement or contract for an interaction between modules in a modular product architecture. Types of interfaces define the various interactions modules can have with each other including attachment, transfer, control and communication, spatial, field, environment, and user. Interfaces are assigned a design risk and design importance to analyze the priority of the modules within the system.

Documentation and analysis of the resultant product architecture is the final step. Design for Manufacturing/Assembly DFMA (Boothroyd Dewhurst, Inc. 2012) and other techniques are used to evaluate the architecture’s robustness. Typically, modification and iterations are required. Once complete, the results are reported as module and interface specifications as well as the Product Management Map.

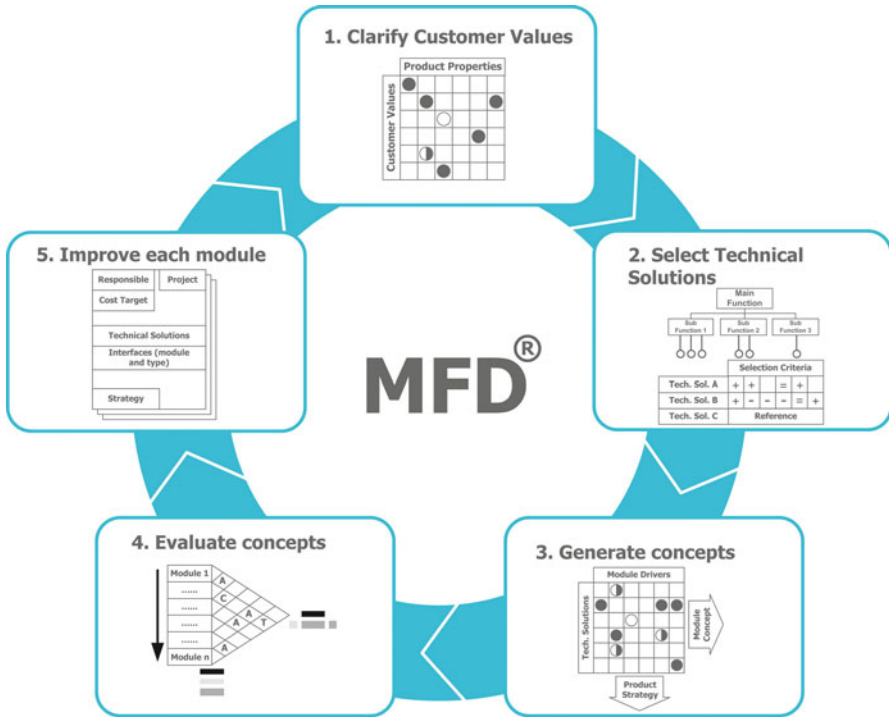


Fig. 24.2 Modular function deployment (MFD) process

### 24.1.1 Quality Function Deployment

The Quality Function Deployment (Akao and Mizuno 1994) used in Modular Function Deployment is focused as compared to a full House of Quality, or HOQ (Hauser and Clausing 1988). Figure 24.3 shows a QFD as it appears in MFD. Customer Values talk about what experience is expected with the product. Product Properties say how the experience will be achieved in the product.

In addition to using focused matrices, MFD relies on strict definitions to ensure data is categorized correctly. A market segment is a group of customers that seek similar Customer Benefits from a product or service. The collection of market segments provides an explanation to the variation in how individual customers behave. Customer Benefits are used to derive Customer Values, which are statements of the experience the customer desires in their use of the product, formulated as if they were in fact spoken by a customer.

Product Properties are measurable, controllable, and solution-free statements about the level to which a function is being carried out. Product Properties are defined in such a way that a method of measurement can be conceived. For example, it is possible to use a voltmeter to measure the voltage of a battery. Therefore, battery



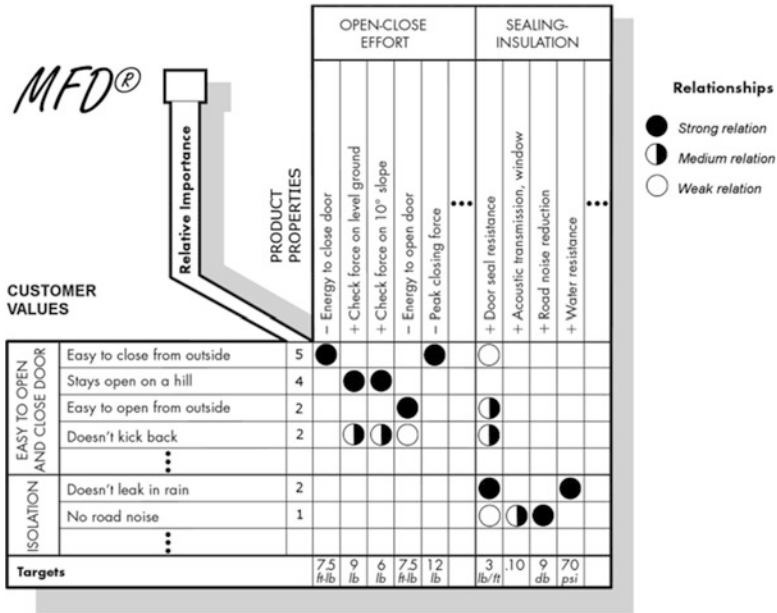


Fig. 24.3 QFD as used in MFD

voltage is a Product Property. The number of batteries, however, is not a Product Property. While it is possible to measure and control the number of batteries, different battery chemistries would change the voltage per battery cell. The number of batteries is dependent on a specific solution and is therefore not solution free.

HOQ features several additional sets of data related to benchmarking as well as the roof which is used to describe trade-offs between different performance levels. For example, a higher value for door seal resistance improves the chances of keeping rain out but makes the door harder to close. MFD does not incorporate the roof. Instead, it uses mechanisms such as functional disaggregation of technical solutions for the trade-offs. By removing this information from the QFD, data is focused on the product. The benefit is a clear understanding of Voice of Customer. Borjesson and Jiran (2012) offer a full discussion on the differences between HOQ and the QFD used in MFD. Interested readers are encouraged to visit the website and download the full article.

### 24.1.2 Design Property Matrix

Design Property Matrix captures the Voice of Engineering by relating Product Properties to technical solutions. The relationship creates the link between the engineering reality and the customer perception defined in the Quality Function Deployment. Product Properties are the key to linking these concepts.

Technical solutions are the physical embodiments of functions needed in the product. It is possible for a function to be fulfilled by multiple technical solutions. For example, to generate suction a rotating impeller may be used. Impellers rotating at high speed tend to be noisy, while small impellers have poor efficiency. An alternative is a tangential fan which is often used in split-unit type air conditions because they operate silently. Although tangential fans deliver good flow rates, they do not produce enough pressure. The selection of a specific technical solution for a handheld vacuum cleaner involves trade-offs between flow, pressure, and noise. A concept selection would consider these trade-offs as well as cost, risk, supply chain issues, and other company-specific requirements to identify the appropriate technical solution for the modular product architecture. Selection of Product Properties is explored in Borjesson (2009).

### **24.1.3 Module Indication Matrix**

The Module Indication Matrix is a unique feature of Modular Function Deployment. By linking technical solutions to one or several of the 12 predefined module drivers within the matrix, company-specific strategies are incorporated into the product architecture. Module drivers describe the strategic intent as it relates to technical solutions or modules. In MFD, there is no direct relation between Customer Values and module drivers that allows the specific company strategy to be applied to the market. The Module Indication Matrix is used to relate company strategy to technical solutions which are, in turn, related to the specific properties and experience delivered by the product.

Treacy and Wiersema (1997) described company strategy using three Value Disciplines, Operational Excellence, Customer Intimacy, and Product Leadership. Operational Excellence is about delivering a product efficiently at the lowest price. Customer Intimacy implies adapting the product to give individual customers exactly what they want. Product Leadership means supplying a product that is better than the competition. Table 24.1 summarizes the module drivers as they relate to a Value Discipline.

Readers familiar with Design Structure Matrix, DSM (Steward 1981), will note that a normal DSM has no features for capturing the company-specific strategy. Attempts have been made to address this issue. Although multiple matrices can be used to achieve similar results to Modular Function Deployment's MIM, the DSM clustering stage becomes more complicated (Blackenfelt 2001).

## **24.2 Cordless Handheld Vacuum Case Study**

For this case study the Black & Decker DustBuster<sup>®</sup> has been selected because it has a medium level of complexity with around 60 technical solutions. Handheld vacuums are well known, inexpensive to purchase, easy to take apart, and have a

**Table 24.1** Twelve module drivers used in MFD

Value discipline	Module driver	Refers to
Operational excellence	Carry-over	Part of a product, or a subsystem of a product, that can be reused
	Common unit	Part and subfunction that can be used throughout the entire assortment of products
	Process/organization	Part that uses a scarce production or development resource or where there are organizational reasons for separation
	Separate testability	Part with a function that may be tested separately before final assembly
	Strategic supplier available	Part that may be developed and sourced from an external strategic partner, typically also affording a reduction of logistical costs
	Recycling	Part that contains a material that will be separated before scrapping at product end of life
Customer intimacy	Technical specification	Part with a performance-driving attribute that is varied to meet different customer values
	Styling	Part that is influenced by trends, fashion, or brand differentiation in such a way that shape or surface material characteristics vary
	Service/maintenance	Part that will simplify service repair if it is easily detachable
	Upgrading	Part that may be changed after the initial purchase to achieve a different level of product performance
Product leadership	Technology push	Part, or a subsystem, that is likely to go through a technology shift during its life cycle as a result of expected, radically changing, customer values
	Planned design change	Part that carries attributes that will be changed according to a product plan

good range of performance and feature variation which lends itself to a useful discussion about architecture. DustBuster, in particular, shows signs of modularity. An exploded view of the DustBuster is shown in Fig. 24.4.

### 24.2.1 Specifications

Development of a modular product architecture begins with a set of baseline expectations for the product. These include the strategic intent of the product, product features, performance levels, required documentation, and an understanding of external relationships with vendors to name a few. Below is the specification for the modularization of the cordless handheld vacuum.

The product will store energy in rechargeable batteries and use a standard DC motor to drive a high-speed rotating impeller which, in turn, creates suction. The air stream gets cleaned by a mechanical filter and the separated debris is stored in a

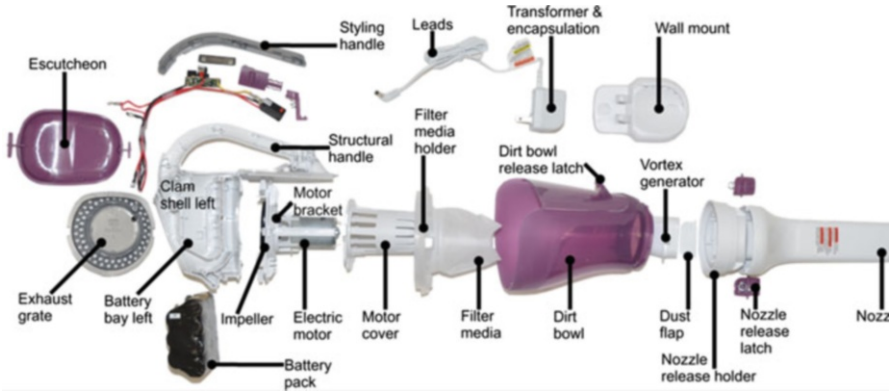


Fig. 24.4 Exploded view of the 12 V unit

container, which is incorporated into the unit. A range of different powers are planned as well as several different sets of attachments and styling.

Sketches or renderings of the new product, as well as a conceptual definition of the interfaces, are required. These interfaces are to be standardized. A conceptual definition of an interface does not include complete documentation in CAD but rather a set of requirements of the interface itself. The physical manifestations of these modules, referred to as module variants, are to be defined as well.

The battery supplier can deliver a complete battery pack and has made investments in automated machinery that can produce battery packs in large volumes at low cost, as long as the battery pack has the same spatial envelope and the battery terminals are in the same location. The battery pack simplifies assembly with no need to deal with individual cells and no quality concerns with poor solders. The battery packs also come pretested from the supplier.

Finally, the impeller design is to remain unchanged. The motor supplier can ship the motor with impeller attached to the shaft. Aligning the impeller on the motor shaft requires a special machine. There are no plans to shift this capability in house because the additional cost of getting the impeller pre-mounted on the shaft is marginal.

### 24.2.2 Market Segments

Market segmentation is not a uniformly understood concept. At many companies, market segments are based solely on consumer attitudes, demographics, or simply a reflection of the range of applications to which a product is sold. It is difficult to translate segments like these into unique and broad Customer Value Ratings. Customer needs-based market segments are required to perform a successful MFD because they describe the variation in the benefits customers seek from the

product. When market segments are based on the merits customers are seeking from a product, the translation becomes clear and precise. The cordless handheld vacuum architecture is based on three target market segments.

#### **24.2.2.1 Family Felicia**

Felicia buys a handheld cordless vacuum to clean up after the family breakfast. Felicia has two kids under the age of five. They make a big mess with their cereals and toast in the morning. Although the ideal might be a wet-and-dry unit to pick up things like milk-soaked cereal, she wants a dry-only unit. In her experience, the wet-and-dry needs to be cleaned frequently and Felicia does not want that hassle. Power has to be sufficient for medium-density particles like small pieces of bread. She sometimes has food stuck in the sofa so the brush attachment is important. After about a year, Felicia understands she may have to buy a new unit, but that is okay provided the price tag is not above \$40. She wants the unit to be ready to work for her when it is needed and appreciates a good run time. Felicia is afraid to provoke the onset of a dust allergy in any of her kids, so an advanced filter would be nice.

#### **24.2.2.2 Danny Do-It-Yourself**

Danny does not have kids so the cordless vacuum he is looking for is intended to pick up stuff in his shop. He primarily does woodwork, but on occasion there are metallic fragments. The highest possible suction power is important. Run time is less important as it generally only takes Danny about 2 min to clean up, well below the typical 10-min run time of most units. The filter has to be able to resist heavy particles without breaking. No wet pickup is envisioned. Danny likes high-tech gadgets, so a rechargeable lithium-ion battery would be appreciated. Alternatively, a battery he can replace, like the one on his VersaPak screwdriver, would be fine. He is not overly sensitive to price and is willing to spend up to \$80 if the unit fills his needs. Danny likes cool features like a charge-in-progress indicator and a charge-complete indicator. His shaver shows remaining run time so that would definitely be a plus on the handheld vacuum. Long product life really is not that important. Features that extend the life, such as replaceable batteries, would be.

#### **24.2.2.3 Sophia Student**

Sophia is in college and there is a surprising amount of lint on the floor in her room. Lint is low density so she is looking for a low-cost unit, even if that means it is weaker. Vertical storage is important and the unit needs to be wall mounted. Although Sophia certainly would never spill liquids on the floor of her room, she has heard some of her friends sometimes have that issue. For this reason, a wet-and-dry capability would be a plus. Unless it drives the price tag up by \$10 in which case

she can just use paper towels. Sophia expects the lifetime of her product to be pretty much the duration of her college stay. Because the walls in her dorm are thin, she appreciates a low-noise unit.

### ***24.2.3 Customer Value Rating***

Standardization tries to find one product which fits all market segments reasonably well. Modularization on the other hand generates a range of products that can be efficiently developed, marketed, produced, and delivered to match the needs of any given segment as closely as possible. Having established three market segments, it is necessary to determine where their differences lie and understand what range of products is required to meet all of their needs.

Customer Value Rating (CVR) documents Customer Value importance for each of the target market segments. A common set of Customer Values are used across all segments, but the importance of each value is segment specific. A numerical scale of 1–5 is used, where 1 is low and 5 is high importance. The advantage of using a numerical scale is that mathematical operations can be performed to predict where there is differentiation between market segments.

### ***24.2.4 Quality Function Deployment***

A QFD relates Customer Values to Product Properties. QFD implies there is a direction of cause and effect. A customer's experience with a product will change as the goal value of a Product Property is changed. Consider the Customer Value Powerful Suction. Altering the value of battery voltage changes the customer's perception of powerful suction. Therefore, there is a relationship between this Customer Value and Product Property.

For the purposes of analysis, these relationships are translated into numbers. A weak relation receives a one, a medium is a three, and a strong gets a nine. No relation is a zero. The use of a 9/3/1/0 scale is based on project experience from Japan (Akao and Mizuno 1994). To get definitive results, strong relations required more emphasis than what a linear scheme of 3/2/1/0 would give. As shown in the legend of Fig. 24.3, circles with three levels of shading are used to indicate the strength of the relation. Figure 24.5 shows a complete QFD for the cordless handheld vacuum.

### ***24.2.5 Design Property Matrix***

The Design Property Matrix (DPM) is the Voice of Engineering. DPM relates the technical solutions needed to fulfill a Function to the Product Properties used to

**Quality Function Deployment**

Segments		Customer Values		Product Properties	
Sophia Student	4 3 1	Easy to take apart to clean	4 3 1	Transparency of bowl	
Danny Do-It-Yourself	3 3 2	Easy to charge	4 4 2	Impact resistance (J)	
Family Felicia	4 4 2	Know it is charging	3 5 3	Dirt bowl volume	
	3 5 3	Long product life	4 2 4	Centrifugal action	
	4 2 4	Minimize handle vibration	4 2 5	Wall space required	
	4 2 5	Easy to clean filter	5 4 3	Off-unit	
	5 4 3	Pick up liquids	3 5 1	Length of cable	
	3 5 1	Always ready to work	5 4 2	Mains voltage	
	5 4 2	Do big cleanups	3 1 5	Noise level	
	4 2 3	Powerful suction	5 3 3	Nozzle height	
	3 1 5	Comfortable to use	4 3 1	Nozzle width	
	5 3 1	Acceptable noise level	4 3 3	Pressure drop	
	3 5 3	Does not break when I drop	5 3 4	On-unit	
	4 3 1	Distinct movement of power switch	5 3 4	Water resistance	
	4 4 3	Cover large areas easily	5 3 4	Particle size	
	5 3 2	No particles into environment	5 3 3	Force to slide button	
	5 3 4	Know when to empty bowl	5 3 5	Force to insert	
	5 3 3	Reach into tight spaces		Vibration damping	
	4 3 3	No filter clogging		Unit weight	
	5 5 5	Works in my home		Dirt pickup capacity	
	5 1 5	Store on wall		Battery charge (mAh)	
				Battery voltage	
				Force to release	
				Charge current (mA)	
				Protect battery from overcharge	
				Indicate charge in progress	

Fig. 24.5 QFD for cordless hand vacuum

quantify those functions. The generation and selection of technical solutions can be done using either a top-down or bottom-up approach. For completely new products or where there is a high innovation content, a top-down function-means tree is the preferred tool.

Roughly 80 % of all projects are a redesign of existing products where the goal is to achieve incremental improvements such as a lower material cost, new features, or improved styling. In these cases, technical solutions may be generated using a bottom-up approach. This lists the technical solutions of several products which cover the scope of the new modular product architecture.

The level of decomposition of the technical solutions needs to be sufficient such that the modules are not predetermined by previous design or bias. The technical solutions of the cordless handheld vacuum are shown in Fig. 24.4. The graphic shows battery pack as a single technical solution. However, the results of the CVR and QFD indicate that battery voltage needs to vary. To achieve this level of resolution the battery pack needs to be decomposed to the individual battery cell, terminal connectors, soldered leads, and an enclosure. Conversely there is no reason to decompose the electrical motor into rotor, stator, brush, bearing, shaft, etc. Strategically, the motor is a purchased component which is being supplied by a preferred vendor.

### **24.2.6 *Module Indication Matrix***

The MIM assigned the company-specific strategy to the technical solutions selected. With only 12 predefined module drivers, it may be hard to conceive a company-specific strategy. As each of the drivers can be applied on individual technical solutions, the number of combinations is astronomical. Since the drivers are generic a specific definition can be used to develop a company-specific application. For example, Process/Organization is used to protect scarce resources. In a company that makes vacuum cleaners, that could be competence in the area of computational fluid dynamics (CFD). This analysis requires software licenses, which may be limited within the organization. In other cases, Process/Organization can be a reference to a production process which is capital or know-how intensive. Such processes benefit from technical solutions with these requirements being collected into a single module.

Module Indication Matrix is scored with the mind-set that each technical solution has at most one strong primary driver and perhaps a weaker secondary driver. Consider the battery. Since battery voltage needs to change, it would be reasonable to put a strong score on Technical Specification to document this strategy.

There is an interaction between the MIM scoring and the concept selections, because in the end, they must be consistent. If it has been decided that a single DC motor covers the entire voltage range from 7.2 V up to 16.8 V, the motor is a common unit. If two different motors are required to cover the voltage range, the motor would be Technical Specification.



Styling is used to indicate technical solutions that get used to create visual differentiation between models. All the buttons are styled, as is the styling handle and escutcheon. The dustbin is color matched, so that it might carry styling as a secondary driver. The styling driver is only applied for something that requires variance in the styling.

A key strategic decision is whether the handheld unit has a consumer replaceable battery. If the consumer is able to purchase the battery pack and replace it, then removing the battery has to be easy and foolproof. Prior business experience has shown this was not profitable. Therefore, the planned architecture will not support a battery pack that is serviceable. As a result, the Service/Maintenance module driver will not be applied.

### ***24.2.7 Module Generation***

To generate modules, a statistical tool called hierarchical clustering is applied to the MFD data. It is possible to cluster only the DPM and then examine the clusters generated to determine if the module drivers are compatible within the clusters. Alternatively, one can cluster with regard to DPM and MIM at the same time. Hierarchical clustering looks at scoring patterns and places items with similar patterns into groups. Two or more groups may cluster into a bigger group. The groups and groups of groups form a hierarchy.

Hierarchical clustering (Romesburg 2004) is generally supported by statistical software packages such as SPSS (SPSS 2010), JMP (SAS 2012), and Minitab (Minitab 2012). The output of hierarchical clustering can be shown in many different ways. A useful and visual representation is the dendrogram. A dendrogram of DPM and MIM is shown in Fig. 24.6.

The vertical line has been drawn in a place intersecting the dendrogram to indicate 21 potential modules. The process of determining the appropriate number of modules is not discussed in this chapter.

The proposed modules, as predicted by the hierarchical clustering and displayed in the dendrogram, are usually different from what was expected. In some cases, the modules might be geometrically difficult to realize because technical solutions are far apart or not connected by anything in particular. When that is the case, the merits of the new structure must be balanced against the difficulty of creating a physical design that allows the technical solutions to be part of the same module.

### ***24.2.8 Conceptual Modules***

Clustering predicts the conceptual modules. A conceptual module is a cluster of technical solutions that have some justification for being considered a module be it a strong module driver score, strong Product Property score, or both. A conceptual module can lead to a product innovation so an open mind and time for reflection are required.

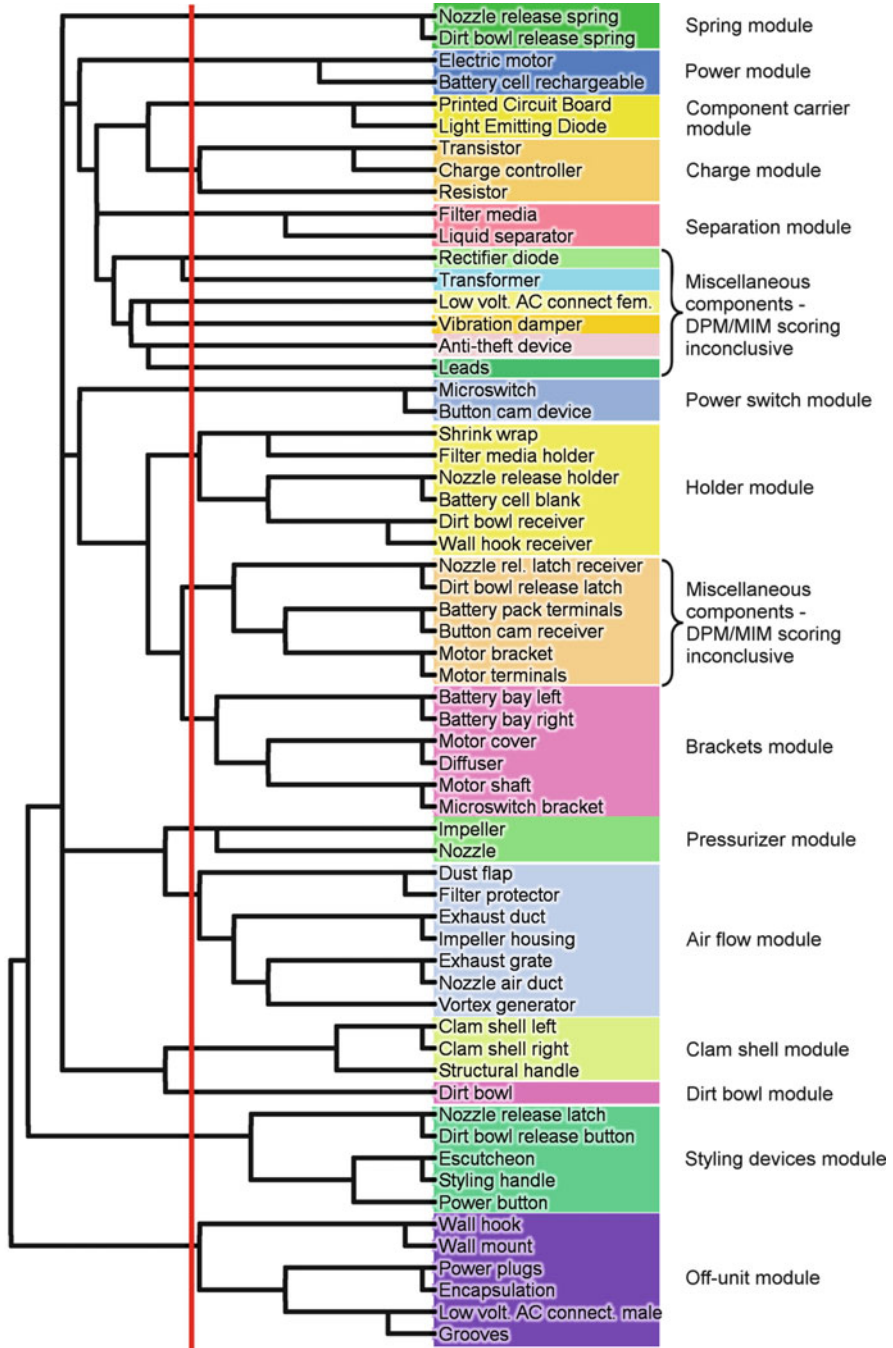


Fig. 24.6 Dendrogram indicating 21 modules



charge module based on a simple rectifier and resistor is of very low cost and reduces battery life because the circuit keeps pushing charge into the battery even when it is full. A charge controller can charge at a high current and turn off charge completely when the battery is full. This extends battery life but costs a bit more. Charge controller is mandatory for lithium batteries but optional for NiCd/NiMH batteries. The need to protect from overcharge is the Product Property that drives the variation on this module.

On wet units, the liquid separator replaces the filter media. Liquid separator and filter media interface the same way to the dirt bowl or the filter holder. Future variants of the separation module may be needed to achieve HEPA rating as it is relevant for a segment of dust-sensitive consumers.

The dirt bowl can be offered in transparent or nontransparent versions. As long as the interfaces to the suction unit and the nozzle are kept stable, the dirt bowl could conceivably be offered in different volumes by allowing it to grow longer. A longer dirt bowl incurs a slightly additional pressure drop which reduces pickup capability, so that must be considered when bowl volume is selected.

All the components changed to achieve a different styling are focused in the styling devices module. It is possible the escutcheon and styling handle make a single geometrical part. Buttons need to move to be useful. They could be allowed to move within a constrained space, allowing the styling devices module to be dealt with conveniently as one piece during assembly.

To improve readability the Product Management Map (PMM) can be sorted in the sequence determined by the dendrogram and color coded as shown in Fig. 24.6. A filtered version of the PMM is shown in Fig. 24.7. A filtered PMM is one where a few modules are selected and only the related technical solutions, Product Properties, and Customer Values are shown.

### 24.3 Modular Launch Planning

Once Modular Function Deployment has defined a modular product architecture, product launch planning can begin. A famous example of product launch planning is the Sony Handycam study by Sanchez (1991), to be described further in Sanchez (forthcoming). Figure 24.8 depicts the development plans for the Sony Handycam range of handheld video cameras. Sony managed to launch a new model on average every 6 months and stay ahead of competitors with this launch plan. The Handycam platform which came out in 1985 was not hampered by the limitations of a legacy platform.

Two of the module drivers relate directly to product launch planning, Technology Push and Planned Design Change. Technology Push is when the impetus for change comes from outside the organization in the form of new technologies that must be incorporated. Planned Design Change is the internally driven desire to change performance.

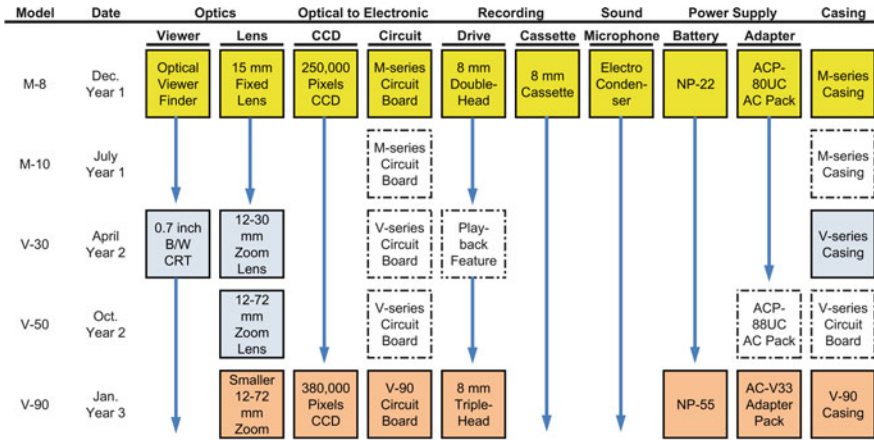


Fig. 24.8 Sony Handycam was improved on average every 6 months (© Ron Sanchez 1991), reprinted with permission

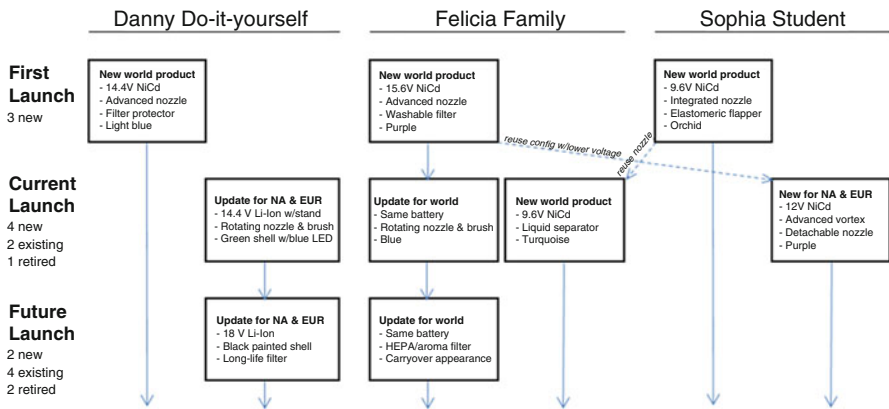


Fig. 24.9 A product launch plan for cordless handheld vacuum

Digital cameras are a good example of Technology Push. A fine consumer SLR camera might have 24 megapixels in 2012, but in 2013, high-end consumer cameras will likely need 36 megapixels to compete. It is possible for a company to manufacture digital cameras without also manufacturing the charge-coupled device (CCD). The CCD is purchased from a CCD manufacturer which controls the technology. So the Technology Push comes from the makers of the CCDs. To enable this strategy, an interface capable of reading CCD signals from higher resolution circuits is reserved, minimizing the need for change for the improved CCD.

Figure 24.9 shows the modular launch plan for the handheld vacuum cleaner. In the initial launch, a product specified for each of the three global market segments is

included. In the second launch phase, only two of the original three products will be available to market. Felicia's 15.6 V NiCd product will be reconfigured using the same battery but updated with a rotating nozzle and brush and a new blue color. Also, three new products will be launched. In the final launch wave, four of the current products will remain with two additional products being made available for Danny and Felicia. Both of these seemingly new products are reconfigurations of existing products that are being taken off the market.

Even without a modular product architecture, one could still do product launch planning. However, every new launch may require significant redesign. This consumes resources, time, cost, etc. Companies are often poor at product launch planning. Plans rarely extend beyond 2 years and changes are very reactive, driven by what competitors do, and not proactive. Modular product architecture enables proactive product launch planning.

## 24.4 Concluding Remarks

Modular Function Deployment is a structured method used to define modular product architectures by integrating Voice of Customer, Engineering, and Company. Each of these voices impact the structure of the resultant modular product architecture. Segmentation of the market and use of a focused QFD enable Voice of Customer. This voice is transformed into the Voice of Engineering by selecting and documenting the necessary technology required to meet the market needs.

Inclusion of strategic intent is a feature that differentiates MFD from other product architecture development methods. Including Voice of Company helps a business to place their product within a market as well as to have the product reflect the strategic position of the organization.

Capturing this information within a Product Management Map allows for iterations throughout the lifetime of the product platform. It also reminds the business what the intent of the architecture was and what trade-off decisions were made to build the architecture. This documentation helps for governance of the architecture to ensure it does not erode over time.

As seen with the application of Modular Function Deployment to the DustBuster, developing an architecture is not a trivial task. Cross functional input is needed to have an architecture that is acceptable to each of the functions within an organization. By setting up this framework as the first step in the product development process will reap benefits for years.

**Acknowledgments** The author wishes to thank his colleagues at Modular Management USA, Inc., who patiently provided feedback on several iterations of this chapter.

## References

- Akao Y, Mizuno S (1994) The customer-driven approach to quality planning and development. Asian Productivity Organization, Tokyo
- Blackenfelt M (2001) Managing complexity by product modularisation, TRITA-MMK 2001:1, ISSN 1400-1179, ISRN KTH/MMK/R-01/1-SE. Doctoral Thesis, Department of Machine Design, Royal Institute of Technology, Stockholm, Sweden
- Boothroyd Dewhurst, Inc. (2012) Company information downloaded on 20 July 2012. <http://www.dfma.com/>
- Borjesson F (2009) Improved output in modular function deployment using heuristics. In: Proceedings of the 17th international conference on engineering design (ICED '09), vol 4, Stanford, pp 1-12. ISBN:9-781904-670087
- Borjesson F, Jiran S (2012) Modular Function Deployment<sup>®</sup> Concepts: QFD. <http://modularmanagement.com/publications/articles>, downloaded on 26 Nov 2012
- Dobberfuhr A, Lange MW (2009) Interfaces per module is there an ideal number? In: Proceedings of the ASME 2009 international design engineering technical conferences & computers and information in engineering conference IDETC/CIE 2009, Aug 30 – Sept 2, San Diego, CA, DETC2009-86872
- Erixon G (1998) Modular function deployment – a method for product modularisation. PhD thesis, The Royal Institute of Technology, Stockholm
- Hauser JR, Clausing D (1988) The house of quality. The Harvard Business Review, May–June, no 3, pp 63–73
- Hölldt K, Tang V, Seering WP (2003) Modularizing product architectures using dendrograms. In: Proceedings of international conference on engineering design, August 19–21, Stockholm, Sweden
- Lange MWL (2012) Personal communication
- Minitab (2012) Company information downloaded on 20 July 2012. <http://www.minitab.com>
- Modular Management (2012a) Interview with MFD founders Dr. Gunnar Erixon, Mr. Alex von Yxkull, and Mr. Arne Erlandsson, personal communication
- Modular Management (2012b) Case “Dynamac”. <http://modularmanagement.com>
- Modular Management (2012c) Case “Esab”. <http://modularmanagement.com>
- Modular Management (2012d) Case “Trane”. <http://modularmanagement.com>
- Modular Management (2012e) Case “MTS”. <http://modularmanagement.com>
- Nilsson P, Erixon G (1998) The chart of modular function deployment. In: Proceedings of 4th workshop on product structuring, Delft University of Technology, Delft, The Netherlands
- Romesburg HC (2004) Cluster analysis for researchers. Lulu, Raleigh, NC. ISBN 978-1411606173
- Sanchez RA (1991) Strategic flexibility, real options, and product-based strategy. PhD dissertation, Massachusetts Institute of Technology. <http://hdl.handle.net/1721.1/13205>
- Sanchez R (forthcoming) Modularity: strategy, organization, and knowledge management. Oxford University Press, Oxford
- SAS (2012) Company information downloaded on 20 July 2012. <http://www jmp.com>
- SPSS (2010) SPSS 12.0. <http://www.spss.com/>
- Stake RB (2000) Using cluster analysis to support the generation of modular concepts in the MFD-method. In: International CIRP manufacturing systems seminar, June 5–7, Stockholm, Sweden
- Steward DT (1981) The design structure system: a method for managing the design of complex systems. IEEE Trans Eng Manag 28(3):71–74, ISSN:0018-9391
- Treacy M, Wiersema F (1997) The discipline of market leaders: choose your customers, narrow your focus, dominate your market. Addison-Wesley, Reading, MA. ISBN 978-0201407198

# Chapter 25

## Optimal Commonality Decisions in Multiple Ship Classes

Michael J. Corl, Michael G. Parsons, and Michael Kokkolaras

**Abstract** A methodology is presented for the determination of the Pareto optimal choice of components and elements to make common between two different classes of military vessels. The use of commonality can produce fleet-wide savings in component purchasing, training, spare parts, vessel construction, etc. The methodology presented here determines the optimal commonality decision and designs the vessel classes to maximize the mission performance per average acquisition cost of each vessel class and the total fleet saving achieved by the commonality. A customized evolutionary algorithm is used to determine the resulting discrete Pareto surface. The methodology is illustrated by its application to the design of two ship classes to perform the specific missions of the US Coast Guard's National Security Cutter and Offshore Patrol Cutter. The results show that the methodology is effective and that not all commonality choices produce a net savings.

### 25.1 Introduction

Simpson provides an extensive survey of efforts to develop rational, analytical methods for the definition of platforms within product family design (Simpson 2004). Most of the literature involving methodologies for and applications of product family design focus on products that are manufactured in large quantities,

---

M.J. Corl  
CDR, U.S.C.G., U.S. Coast Guard Academy, New London, CT, USA

M.G. Parsons (✉)  
University of Michigan, Ann Arbor, MI, USA  
e-mail: [parsons@umich.edu](mailto:parsons@umich.edu)

M. Kokkolaras  
McGill University, Montreal, QC, Canada





**Fig. 25.1** National Security Cutter (*left*) and Offshore Patrol Cutter (*right*)

such as hand tools and automobiles (Gonzalez-Zugasti and Otto 2000; Gonzalez-Zugasti et al. 2000; Simpson et al. 2001, 2005; Messac et al. 2002; Nayak et al. 2002; Fellini et al. 2004, 2005, 2006; Fujita and Yoshida 2004).

This chapter focuses on the development and application of product family methods for the public procurement of complex military products that are typically manufactured in very small numbers. This difference in the quantity manufactured changes the way the product value is modeled and optimized. Moreover, the savings that will result from the use of a platform are explicitly considered since not all commonality decisions will result in an overall savings.

The application area for which this methodology was developed and tested is optimal commonality decisions in multiple ship classes (Corl 2007; Corl et al. 2007a, b; Parsons 2009). In ship design, common hull blocks, main engines, engine rooms, ship service electrical generators, sensors, and weapons can be used to provide commonality across multiple ship class variants. Commonality within multiple ship classes can save money through larger bulk procurements of components, learning during production leading to reductions in the required man-hours, reduced spare part procurement and storage, consolidated and reduced training, etc. This must be introduced, however, with an acceptable loss of performance compared to the use of the optimal design developed for each class individually. Military ships are generally designed today to maximize their mission effectiveness without systematic consideration of the detailed design of other ships in the fleet; an exception being compatible communications and weapons control.

The test application utilizes the specific missions of the US Coast Guard's high and medium endurance cutter fleets that include the Maritime Security Cutter Large (WMSL), formerly the National Security Cutter (NSC), and the Maritime Security Cutter Medium (WMSM), formerly known as the Offshore Patrol Cutter (OPC), as shown in the conceptual images of Fig. 25.1. The first NSC was actually launched in September 2007 and the OPC is being designed as of 2012. Table 25.1 shows the approximate design characteristics of both ships (USCG website 2006). The initial mission requirements for these two classes of ships were used in the research described here to examine the validity of the optimization methodology. This approach was used for academic methodology development only and was not intended to be a comparison to US Coast Guard design work for either the NSC or the OPC.

**Table 25.1** Approximate characteristics of US Coast Guard's vessels

Characteristics	NSC	OPC
Number of cutters	8	25
Length overall	127.4 m (418')	Estimate 106.7 m (350')
Maximum beam	16.46 m (54')	Estimate 15.54 m (51')
Navigational draft	6.4 m (21')	Estimate 6.4 m (21')
Displacement	4,368.3 t (4,300 LT)	Estimate 3,047.6 t (3,000 LT)
Sprint speed	28 kts	26.5 kts
Sprint speed range	2,600 nm	1,550 nm
Sprint speed endurance	3.91 days (94 h)	2.5 days (60 h)
Economical speed	8 kts	9 kts
Economical speed range	12,000 nm	9,000 nm
Endurance	60 days	45 days
Propulsion plant	2 diesels, 1 gas turbine	4 main diesel engines
Bow thruster	Yes	Yes
Gun for weapon system	57 mm gun	57 mm gun
Gunfire control	Mk-160/Mk 46/SPQ-9B	Mk-160/Mk 46/SPQ-9B
Operating days away from port	230	230
Mission days/year	200–220	200–220
Berthing capacity limit	148	106
Number of helicopter hangars	2	2

## 25.2 Problem Formulation

The commonality optimization is approached by formulating and solving a multi-objective optimization using three-objective functions:

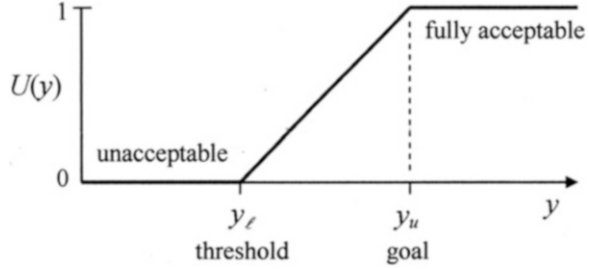
$$f_1(\mathbf{x}_1, \mathbf{x}_C) = (\text{Overall NSC Mission Effectiveness}) / (\text{Ave. NSC Acquisition Cost}) \quad (25.1)$$

$$f_2(\mathbf{x}_2, \mathbf{x}_C) = (\text{Overall OPC Mission Effectiveness}) / (\text{Ave. OPC Acquisition Cost}) \quad (25.2)$$

$$f_3(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_C) = \text{Net Fleet Savings due to Commonality} \quad (25.3)$$

The objective functions  $f_1$  and  $f_2$  maximize the ratio of the individual mission effectiveness to average acquisition cost for the two cutter classes, respectively. The objective function  $f_3$  is the total fleet savings realized through the use of the commonality. The ship design synthesis models used within the evaluation of  $f_1$  and  $f_2$  include the relationships among the independent ship design variables and the many dependent variables necessary to define a preliminary ship design as well as

Fig. 25.2 A typical design requirement fuzzy utility  $U(y)$



many design constraints imposed by physics, sound naval architecture practice, and the customer. The design variables are as follows:

- $\mathbf{x}_1$  = NSC mission design independent variables
- $\mathbf{x}_2$  = OPC mission design independent variables
- $\mathbf{x}_C$  = commonality definition variables

Multi-objective fuzzy optimization is used to solve the problem. In fuzzy optimization, fuzzy membership functions or fuzzy utilities  $0 \leq U(y) \leq 1$  are defined for each criterion or constraint. They represent the degree to which some requirement is satisfied. Each independent variable  $y$  is selected to represent a design aspect appropriately. A typical fuzzy utility, as might be used to express a requirement for ship speed to accomplish a particular mission, is shown in Fig. 25.2. The region with  $U(y) = 0$  is clearly unacceptable to the designer, while the region with  $U(y) = 1$  is fully acceptable. The fuzzy region between the minimum acceptable threshold  $y_\ell$  and the design goal or target  $y_u$  yields a fuzzy quantity value between 0 and 1. The fuzzy transition could be developed by design judgment or from expert opinion using the analytical hierarchy process (Saaty 1996) or similar methods (Ayyub 2002).

If each design goal and constraint is expressed by an appropriate utility function  $U_i(\mathbf{x})$  that depends on the design choices  $\mathbf{x}$ , a fuzzy optimum, using minimum correlation inference (Kosko 1992), is given by the maximization of the optimization criterion (objective function) or total utility  $U(U_i(\mathbf{x}))$ :

$$U^* = \max_x U(U_i(\mathbf{x})) = \max_x [\min_i (U_i(\mathbf{x}))] \tag{25.4}$$

Minimum correlation inference corresponds to an “AND” conjunction where each of the goals and constraints must be met. This seeks the design  $\mathbf{x}$  that maximizes the worst (minimum) satisfaction of any of the applicable goals and constraints  $i$ . This approach yields a multi-objective compromise among all of the conflicting goals and constraints and treats them all in a similar manner. It has the search advantage that there can always be a feasible solution that can be improved.

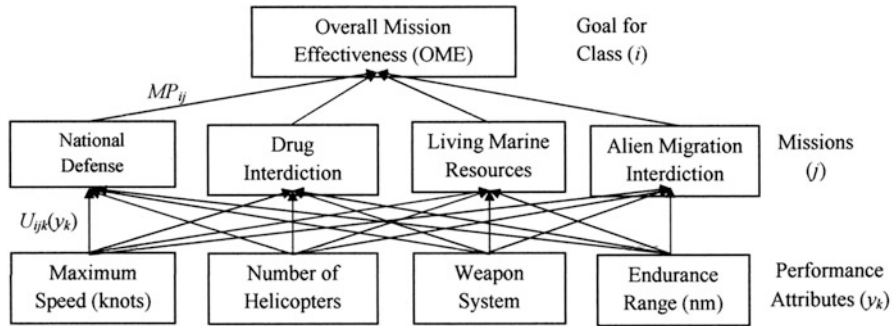


Fig. 25.3 Overall mission effectiveness hierarchy for OPC

### 25.2.1 Mission Effectiveness/Cost Objectives

For  $f_1(\mathbf{x}_1, \mathbf{x}_C)$  and  $f_2(\mathbf{x}_2, \mathbf{x}_C)$ , the ratio of overall mission effectiveness to average acquisition cost for vessel  $i$  is given by

$$f_i = [Performance/Cost]_i = \sum_j MP_{ij} \min_k [U_{ijk}(y_k)] / Cost_i \quad (25.5)$$

The vessel overall mission effectiveness (OME) is modeled similar to Brown and Salcedo (2003). The overall mission effectiveness of the OPC is shown as a hierarchy in Fig. 25.3. The OPC has four missions  $j$  as shown, and the  $MP_{ij}$  are the mission profile percent time each vessel  $i$  will spend on its mission  $j$ . The missions of the vessel classes and the related  $MP_{ij}$  were taken from US Coast Guard planning (USCG internal, USCG 1995). The NSC and OPC missions both include National Defense, Drug Interdiction, and Living Marine Resources (LMR) missions. The OPC also performs USCG nonmonelature used, while the NSC also performs additional General Defense operations in support of the US Navy.

The ability of each ship  $i$  to successfully accomplish each mission  $j$  is assumed to depend upon  $k$  performance attributes  $y_k$ . The contribution of each performance attribute  $y_k$  of ship  $i$  to the success of its mission  $j$  is characterized by a fuzzy membership function or fuzzy utility  $0 \leq U_{ijk}(y_k) \leq 1$ . The overall mission effectiveness is obtained by minimum correlation inference. The  $Cost_i$  is the average acquisition cost of ship  $i$ . The first two objectives are written as benefit/cost ratios so that any overdesign caused by the use of commonality will be penalized as wasteful.

For illustration, the effectiveness of the two vessel classes to perform their missions is assumed to be dependent on the following attributes: maximum ship speed, number of helicopters carried, weapon systems, and endurance range. The fuzzy utilities assumed for the two vessel classes for the Drug Interdiction Mission are shown in Fig. 25.4.

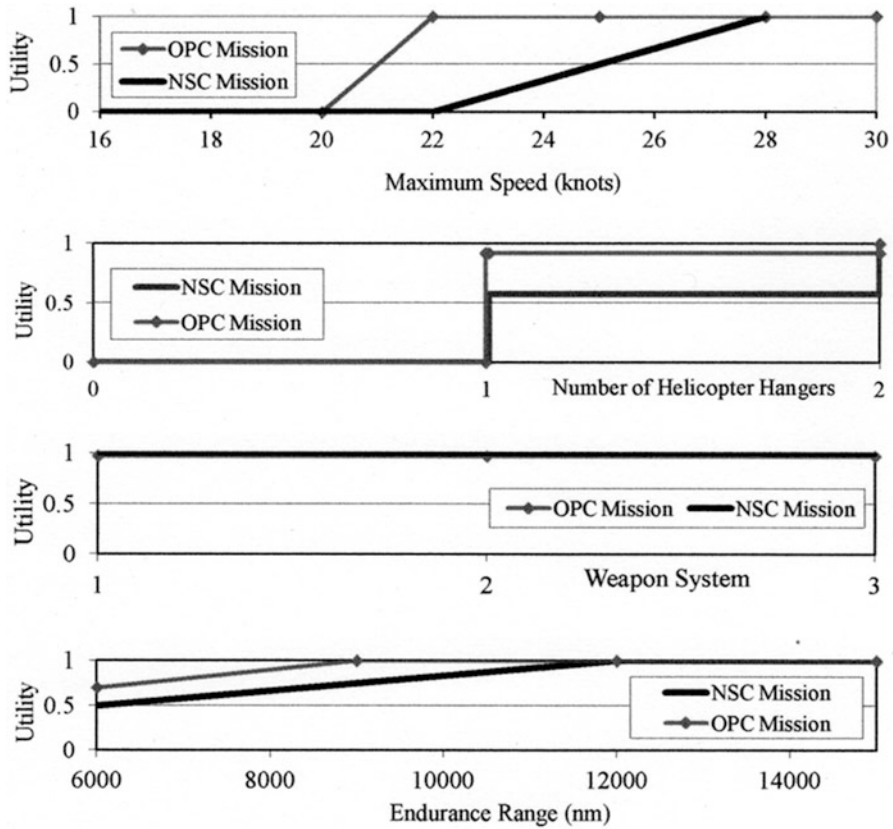


Fig. 25.4 Fuzzy utilities for drug interdiction missions (Corl et al. 2007b)

### 25.2.2 Net Fleet Savings Objective

The net fleet savings objective  $f_3(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_C)$  aggregates the savings in all fleet-wide costs resulting from the use of commonality defined by  $\mathbf{x}_C$ . To validate the methodology, these savings were limited here to those resulting from larger bulk purchases of components and savings from the construction learning curve when more work of the same type is performed. The bulk purchase models were linear with respect to the number of common components purchased up to some assumed percentage reduction (5–15 %) for the maximum number possible (for 33 total vessels).

A typical ship construction labor man-hours learning curve for a learning or experience rate of 0.95 is shown in Fig. 25.5. This learning rate results in the labor man-hours dropping by 5 % every time the number of units constructed doubles. This effect is important where only a small number of units are constructed. Observed shipbuilding learning rates have been as high as 0.80. A conservative

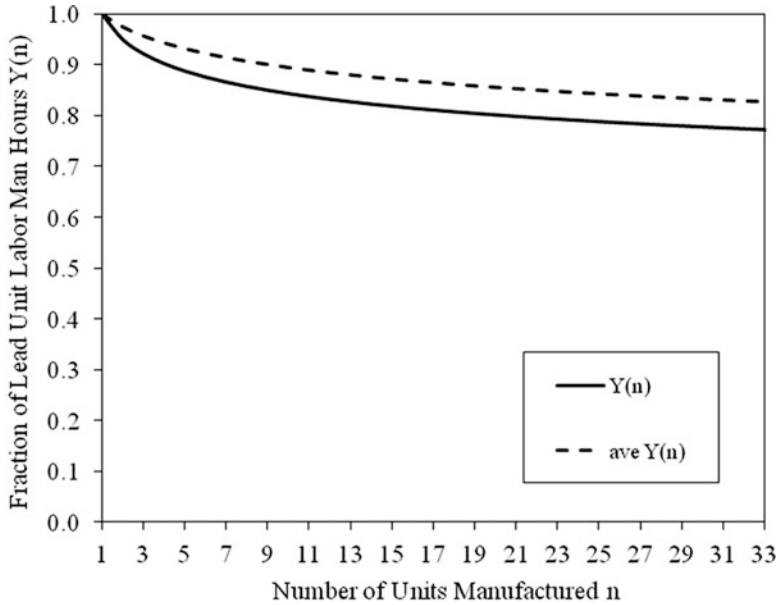


Fig. 25.5 Typical ship construction labor learning curve for learning rate 0.95

rate of 0.95 is assumed here. The global effect of commonality on the cost of the entire fleet of ships involving the 8 NSC mission design vessels and 25 OPC mission design vessels is used.

### 25.2.3 Design Variables

The ship class design variables  $\mathbf{x}_1$  and  $\mathbf{x}_2$  include the limited number of independent variables required to define a consistent and feasible design within the ship synthesis model described below.

The commonality variables  $\mathbf{x}_C$  contain an additional set of integers that specify whether a particular component will not be constrained to be common (0) or if one of its options (1, . . . , n) will be required to be common between both ship classes. If a given component is designated as common, the synthesis of both ships will be constrained to use that component. Each potential commonality component will have two or three component choices. By varying the number and combinations of the commonality components, the design space will be populated. Potentially common components here will be the weapon systems, diesel ship service electrical generators, diesel cruise engines, the superstructure blocks, and the midship section hull blocks. The various combinations of these commonality components will be used to determine which sets of common components will result in Pareto optimal designs for the NSC mission design and OPC mission design.

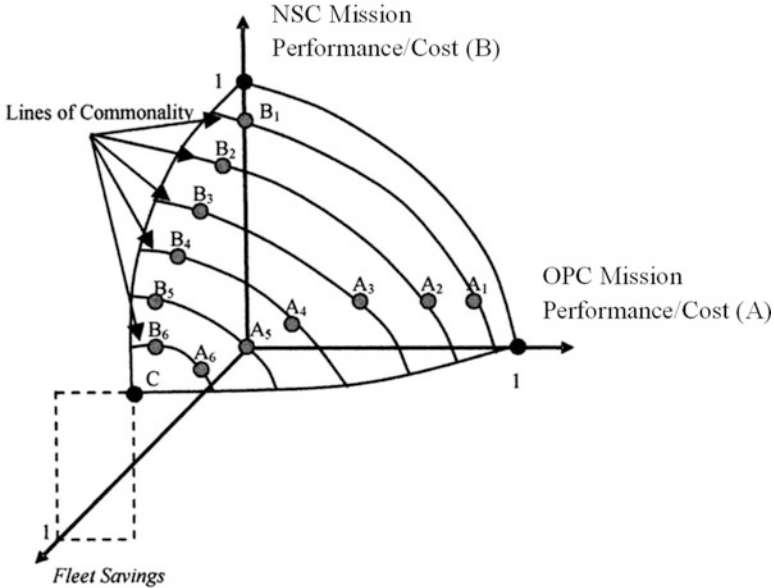


Fig. 25.6 Expected discrete Pareto front (adapted from Corl 2007)

#### 25.2.4 Expected Discrete Pareto Front

As the various combinations of commonality are applied to the designs, the optimization will be repeated to fill out the three-objective Pareto front or Pareto surface. Figure 25.6 shows a schematic of the expected discrete Pareto front that will be obtained for this multi-objective optimization. The mission performance/cost vertical base plane is the Pareto front that would result if a two-objective optimization was performed on  $f_1$  and  $f_2$  for the single ship design that would best perform both missions. The discrete Pareto surface coming out from this base plane defines the solution to the three-objective problem of interest here.

Every set of commonality components  $\ell$  will result in a solution for an OPC mission design  $A_\ell$  and an NSC mission design  $B_\ell$  that will be located on a line of commonality. If a single ship were being considered for both missions, this line would be the two-objective Pareto front for NSC mission design performance/cost and OPC mission design performance/cost. For specific commonalities, Ship  $A_1$  and Ship  $B_1$  might share the NSC mission design's midship section hull blocks, Ships  $A_2$  and  $B_2$  might share the NSC mission design's midship section and cruise engines, and so on. As more things become common among the ships, the savings can increase and the ship designs will tend toward each other on the Pareto surface as more effectiveness is sacrificed for commonality. Once every item on the ship is determined to be common, the result will be one design for both missions. This design is shown as point C in Fig. 25.6. Once every combination of common components is used in the

optimization, the discrete Pareto front will be fully populated. The Pareto front will not be continuous because of the discrete nature of the commonality variable. Rather, the Pareto front will be a collection of discrete points as shown in Fig. 25.6.

### 25.3 Ship Design Synthesis Model

A ship design synthesis model is a parametric model that produces a fully balanced and feasible ship design for use in conceptual or preliminary design trade-off studies and optimization (Parsons 2003). These complex models can have anywhere from a few (5–10) independent design variables to a much larger number of design variables and design choices at the discretion of the designer. The many ship characteristics that depend upon these design variables are computed from first principles or estimated by parametric models derived from past acceptable design practice. These models define dimensions, hull shape characteristics, weights, volumes, propulsion power requirements, electrical loads, etc. based upon the design variables and many related dependent variables. To define a feasible ship, constraints related to adequate ship stability, operational practicality, regulatory requirements, and accepted design practices are also included. The US Navy's Advanced Surface Ship Estimating Tool (ASSET 2005) is used routinely in early design trade-off studies for US surface combatants. ASSET is a combination of first-principle algorithms and regression models based upon historical US combatant ship data.

The feasible, weight- and volume-balanced vessel designs used in this study were synthesized using an adaptation of the US Coast Guard's Performance-Based Cost Model (NSWC Carderock Division 1998), which is used routinely to study the effects of the choice of 21 design variables on early Coast Guard cutter conceptual designs. This synthesis tool was developed by the US Navy using components of its Advanced Surface Ship Evaluation Tool (ASSET 2005) and the Canadian equivalent SHOP5. The related costs are estimated in constant 1998 US dollars using cost estimating relationships (CERs) based upon the US Coast Guard's WHEC 378, WMEC 270, WMEC 210, and Great Lakes Icebreaker. The model is capable of synthesizing frigate-sized, deepwater cutters of over 1,500 metric tons (t) and providing estimates of the acquisition, operational, and support costs. The engines and ship service generators included in these designs come from catalogs of available engine and generator models.

For this optimization, the USCG Performance-Based Cost Model was modified to reduce the number of inputs to the eight as listed in Table 25.2 using typical design practices and models internally to obtain the other inputs usually required from the designer. All ship design constraints needed to ensure a balanced and feasible design are included internal to the synthesis.

The design variables in Table 25.2 compose  $\mathbf{x}_1$  and  $\mathbf{x}_2$  describing the NSC mission design vessel and the OPC mission design vessel, respectively. The ranges considered for these variables were roughly  $\pm 10\%$  from the values for the actual



**Table 25.2** Ship design variables and ranges (Corl et al. 2007b)

Independent variables	Variable range used
Power plant type	1 or 2
Midship coefficient	0.75–0.99
Block coefficient	0.45–0.85
Length	82.3–143.3 m (270'–470')
Maximum speed	19–31 knots
Range at cruising speed	8,000–14,000 nm
Number of helicopter hangars	1 or 2
Weapons system type	1, 2, or 3

NSC and the OPC designs. The power plants considered were (1) a combined two cruise diesel engine and two sprint diesel engine (CODAD) plant or (2) a combined two cruise diesel or one sprint gas turbine (CODOG) plant. Both options have twin propulsion shafts with mechanical gearing and a controllable pitch propeller. The weapon suites were (W1) a 46 mm gun, (W2) a 57 mm gun, and (W3) both a 57 mm gun and a Phalanx close-in weapon system (CIWS).

## 25.4 Evolutionary Optimization

### 25.4.1 Overall Optimization Strategy

The multi-criterion optimization was undertaken in two steps. The two-objective problem involving just  $f_1$  and  $f_2$  was solved first. This provided the vertical base plane solution shown schematically in Fig. 25.6. In addition to validating the problem formulation, modeling, and the basic optimization algorithm, it provided insight into the fundamental design trade-offs between the NSC and OPC mission design requirements. This also provided an understanding of what components and which options to include as candidates for a commonality decision. It is important to limit this candidate list due to the combinatorial nature of this discrete optimal decision. The three-objective problem was then undertaken with a specialized version of the optimization algorithm due to the discrete nature of the final Pareto surface with the need for only two discrete solutions for each commonality possibility.

### 25.4.2 Two-Objective Evolutionary Optimization Method

An evolutionary (real-coded genetic algorithm) optimization process (Goldberg 1989; Michalewicz 1996; Deb 2001; Osyczka 2002) was designed to provide the Pareto front with a diverse set of solutions. The basic optimization process used to

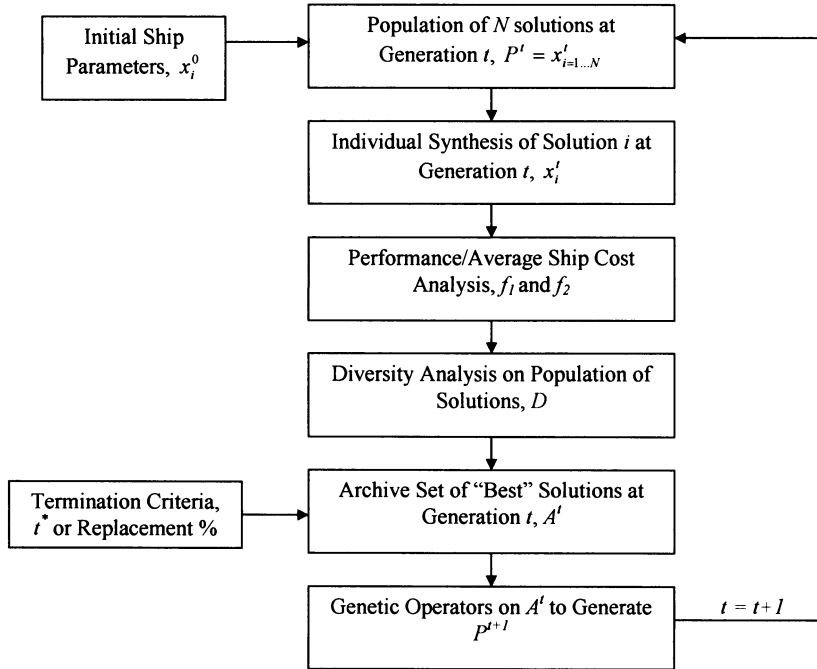


Fig. 25.7 Two-objective evolutionary optimization algorithm

obtain the two-objective solution involving only  $f_1$  and  $f_2$  is illustrated in Fig. 25.7. The details of each portion of this algorithm are described briefly in the following.

The optimization process is a multi-objective evolutionary algorithm based primarily on Zalek’s work (Zalek et al. 2006; Zalek 2007). Penalty functions for constraint satisfaction were not needed since all constraints were implemented within the ship synthesis model. Zalek’s algorithm was based primarily on Deb (2001) and Zitzler et al. (2003) with some original concepts that were developed for the specific nature of his work. Many of the procedures used are standard methodologies in evolutionary algorithms. However, Deb’s influence can be seen in the nondominance sorting algorithm and in the tournament selection method used. The use of an archive as an elitism operator was taken from the Strength Pareto Evolutionary Algorithm (SPEA) work of Zitzler.

The initial population of ships,  $P^0$ , consists of randomly generated combinations of the  $n = 16$  independent variables,  $x_i^j$ . Each set of parameters is input into the ship synthesis model and a ship is developed. Not all combinations of inputs will generate a feasible ship. If a ship is not feasible, a new set of parameters is developed and synthesized. The process continues until the minimum population of ships,  $N$ , has been created.

The population at any given generation  $t$  is set to have a minimum number of ship variants  $N$ . There is no maximum on the number of solutions in the archive.

By allowing the population to grow without a maximum, the variable space is searched more efficiently and effectively. If an artificial criterion were used to limit the size of the population, as done by Zitzler, nondominated solutions may be lost and never recovered. In order to maintain elitism, the population at  $t > 0$  consists of the previous generation's archive and the offspring that are created in the current generation.

In order to ensure that a wide range of solutions are generated along the entire Pareto front, solution diversity is analyzed. The diversity measures a given solution's distance from its nearest neighbor solutions. Using a method similar to that utilized by Zalek (2007), the distance to the nearest three neighbors in the  $n$ -dimensional independent variable space is calculated. The average of the three closest solutions,  $D_{RAW}$ , is calculated, normalized by the maximum value of  $D_{RAW}$  for all solutions in the population, and then maximized to ensure solution spreading along the Pareto front. Using the three nearest neighbors ensures that a solution that has a single close neighbor will not be penalized. By only using the three nearest neighbors, a localized diversity is calculated. Identical solutions are not considered in the calculation of diversity. By eliminating duplicate solutions, the diversity of the population is more easily maintained. Duplicate solutions run the risk of dominating the genetic processes and creating additional duplicates.

The archiving of best solutions serves three important purposes. First, it creates the pool of potential parents for tournament selection and the evolutionary generation of offspring. Second, it allows for the measurement of how much the Pareto front is progressing from one generation to the next. Finally, it serves as the elitism operator for the algorithm. The archive is developed using standard dominance sorting techniques. Each solution in the population is compared to every other solution to check for dominance. Dominance occurs when each of the objective values is not worse than those of the other solutions and at least one objective is better than that of the other solutions.

The archive is generally made up of the nondominated set of solutions. However, during the early generations it may consist of lower ranked solutions in order to reach the minimum number of solutions. As the solutions become more refined, the number of nondominated solutions increases and eliminates the need to carry lower ranked solutions in the archive. There is no maximum to the size of the archive. If the number of solutions on the Pareto front were limited in size, the fine details of the Pareto front might not become apparent. These might include important knuckles or gaps in the front.

The optimization process has two termination conditions. The user can set a maximum number of generations  $t^*$ . A second stopping condition becomes active when the archive becomes stagnant. The solutions in the archive carry markers which indicate if they have been carried over from previous generations or if they are newly generated offspring. If 99 % of the solutions from one archive to the next are the same, the program stops and outputs the final archive.

The archived solutions in  $A$  make up the potential parent solutions in the mating pool. Once the archive has been created, those solutions are compared in a tournament selection process (Michalewicz 1996; Li and Parsons 1998). In tournament

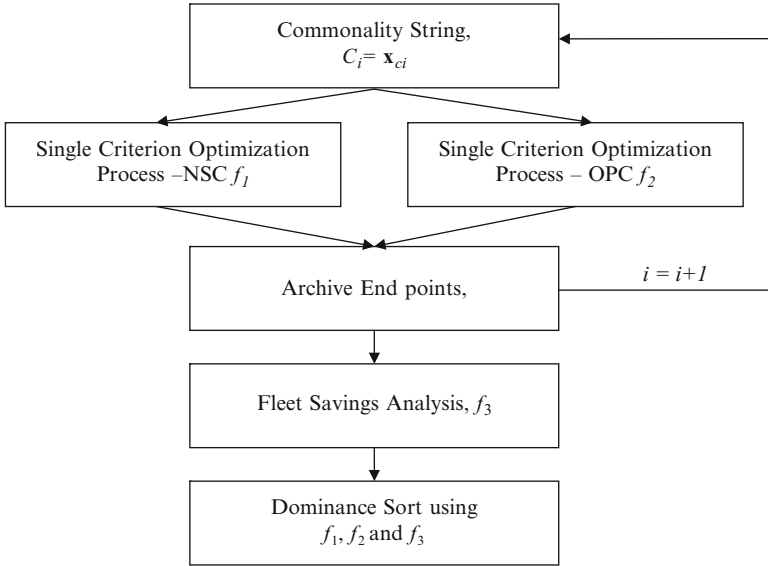
selection, archived solutions are randomly paired together. Each pair of solutions is compared using two tests. The first test asks if either solution dominates the other. If one solution dominates the other, the dominant solution is placed in the mating pool. If neither solution dominates, the diversity measure is added to the objective functions and the solution with the higher sum is selected for the mating pool. In earlier generations, the first test is more likely to determine which solution in a pair will become a parent. As the archive is filled entirely with nondominated solutions, the second test tends to distinguish between the two solutions. In essence, early in the optimization process, the goal is to generate more nondominated solutions. Later in the process the goal shifts to creating more diverse solutions.

The optimization process creates a minimum of  $k$  child solutions per generation. As the archive grows beyond its minimum, more child solutions are developed in proportion to the size of the archive. Arithmetic crossover (Michalewicz 1996) is applied to two randomly selected parents to create an offspring. The optimization process also utilizes a mutation operator that allows for a more global search of the variable space. The mutation is set up similar to Zalek's work in that the rate of mutation starts out small with a large mutation magnitude in the earlier generations. This allows for a broad search of the variable space. As the generations progress toward the termination condition, the rate of mutation is increased exponentially, while the mutation magnitude is decreased exponentially. This allows for a more local search of new design solutions. By searching closer to existing solutions, the optimization process looks to fill in the gaps between existing solutions to create a more refined Pareto front.

### ***25.4.3 Three-Objective Discrete Optimization Process***

The optimization was adapted to incorporate the commonality variable (string) and accommodate the discrete nature of the final Pareto surface with the need for only two discrete solutions for each commonality possibility. The dominance sort was also modified to include all three design objectives. The complete three-objective optimization process can be seen in Fig. 25.8. With the reasonable number of combinations of the possible commonality choices, the commonality choices were searched by an exhaustive search.

The optimization process for the discrete solution was similar to the one seen in Fig. 25.7. A few changes were made, however, in order to make this process more efficient in solving the three-objective problem. The most obvious change is that the optimization is run for the OPC and NSC mission designs independently. Since finding only the end points of the two-objective Pareto front is the goal for each candidate commonality, it is possible to optimize for each objective at higher precision when performed independently. The dominance sorting and tournament selection processes are performed using only the respective objective for each mission ship. This single-objective sorting provides a more efficient search for



**Fig. 25.8** Three-objective discrete optimization process

the best OPC mission design and the best NSC mission design for each commonality string.

In each optimization in which commonality was applied, a fleet savings from the commonality was calculated. The total fleet savings that results from the use of commonality in the designs was calculated by summing up each of the applicable savings associated with each of the common components. The fleet savings for each pair of ship classes was calculated relative to the vertical base plane pair of ship classes that were designed with no commonality.

In order to determine which pairs of these designs were on the discrete Pareto front, another dominance sort was performed. The ships are compared to each other in pairs. Each pair will have an NSC mission design, an OPC mission design, and the associated fleet savings. Dominance is determined by comparing the NSC mission designs using  $f_1$ , the OPC mission designs using  $f_2$ , and each pair's  $f_3$ . The solutions that make up the nondominated set of solutions are the final three-objective discrete Pareto front anticipated in Fig. 25.6.

## 25.5 Sample Results

For testing and evaluation of the methodology, the commonality design for a fleet of 8 NSC mission design vessels and 25 OPC mission design vessels was undertaken. Following the overall optimization strategy, the two-objective optimization was first performed for one ship design to satisfy both the NSC mission requirements

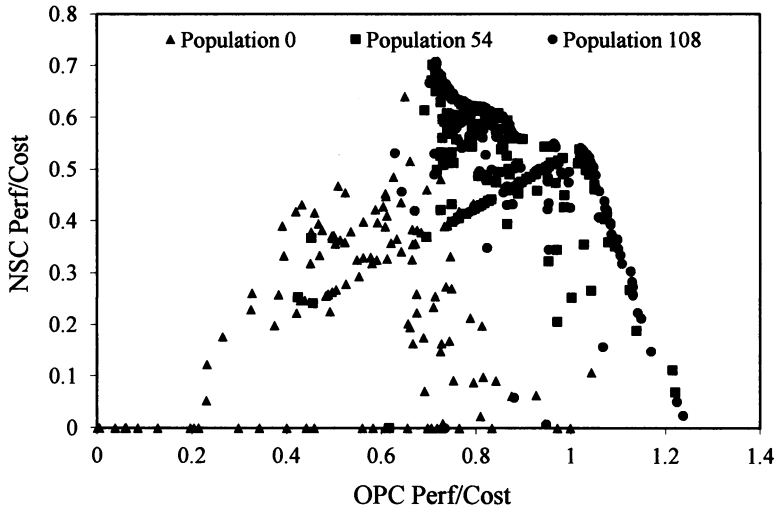


Fig. 25.9 Progression of evolutionary solution toward the Pareto front (Corl et al. 2007b)

and the OPC mission requirements. Recall, this is the vertical base plane in Fig. 25.6. A typical progression of these solutions through 108 generations is illustrated in Fig. 25.9. The dense line through the center of the figure is the dividing line between designs with one helicopter hangar (below and to the right, the less capable OPC end) and two helicopter hangars (above and to the left, the more demanding NSC end).

The final Pareto front for the two-objective optimization for a single design to do both missions was studied to establish which choices of weapons, cruise engines, and ship service diesel generators occurred in the Pareto optimal designs. This was used to guide which of the many options to include in the commonality study. All solutions used the two cruise diesel—two sprint diesel CODAD plant type. The results showed that only two cruise engines (C7 or C9 from the synthesis model catalog of 13 possible engine choices), three ship service generators (G0, G1, or G3 from the possible 12), and two weapon systems (W1 or W3) were Pareto optimal, as shown in Fig. 25.10. These cruise engine and ship service generator results were selected to reduce the scope of the commonality study. It was also noticed that once the number of helicopter hangars was set, there was very little variation in the resulting superstructure volume: either a small superstructure (one helicopter hangar) or a large superstructure (two helicopter hangars). Also, the number of helicopter hangars resulted in little variation in the beam and depth of the hull, with either a small beam and depth or a larger beam and depth. Thus, common superstructure blocks (small or large) and common midship section hull blocks (small or large) were included as commonality options.

The modeling for the commonality was then an integer vector of the form  $\mathbf{x}_C = [0 \ 2 \ 0 \ 1 \ 1]^T$  where the positions indicate the commonality decision for weapons, cruise diesel engines, ship service diesel generator sets, superstructure

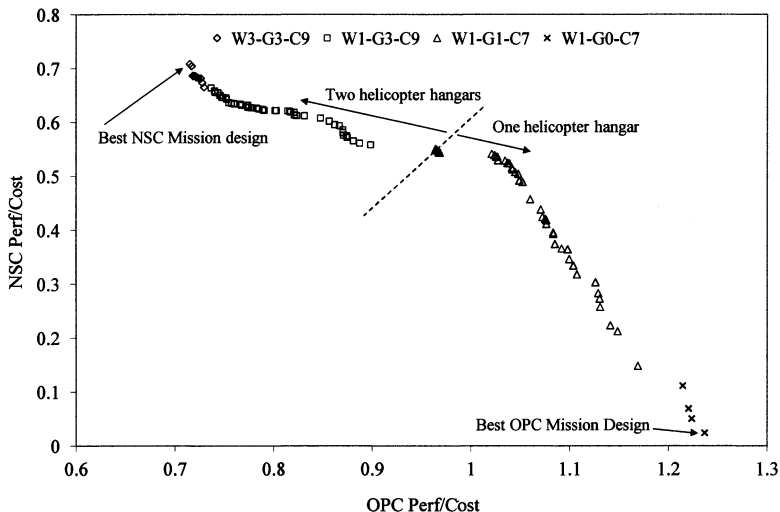


Fig. 25.10 Natural commonality within the Pareto front solutions

(blocks), and midship section hull blocks, respectively; the zero indicates no commonality is imposed; and a nonzero entry indicates the index number of the commonality choice imposed on the designs.

The two-objective evolutionary algorithm was next adapted further to obtain two separate, higher-quality solutions near the ends of the  $f_1$  and  $f_2$  base plane Pareto front, since the end points as shown in Fig. 25.10 were all that was needed. Given the five possible commonality components (weapons systems, cruise engines, generators, superstructure, midship hull blocks) and the choices associated for each (no commonality or a specific commonality), the total number of possible commonality strings was  $4 \times 3 \times 4 \times 3 \times 3 = 432$ . However, some of these had conflicting requirements. For example, ships with the large superstructure could not also have the smaller midship hull blocks, and they required the large cruise engines to make the needed speed.

The three-objective analysis was, therefore, run for the 288 possible feasible combinations of commonality decisions. As shown in Fig. 25.11 in normalized objective function space, these results produced three bands of similar designs: 128 NSC mission design vessels with two helicopter hangars, 160 NSC mission design vessels with one helicopter hangar, and 288 OPC mission design vessels with one helicopter hangar. Of these results, 129 of the pairs resulted in *negative* net fleet savings when more expensive components were being imposed on the 25 less demanding OPC mission design vessels. This is a common fallacy of much of previous work on platforms where it is usually assumed that all commonality is good. Because the optimization criterion used here involves performance/cost and no increase in the performance utilities occurs with more than the goal level, overdesign results in a loss in performance/cost.

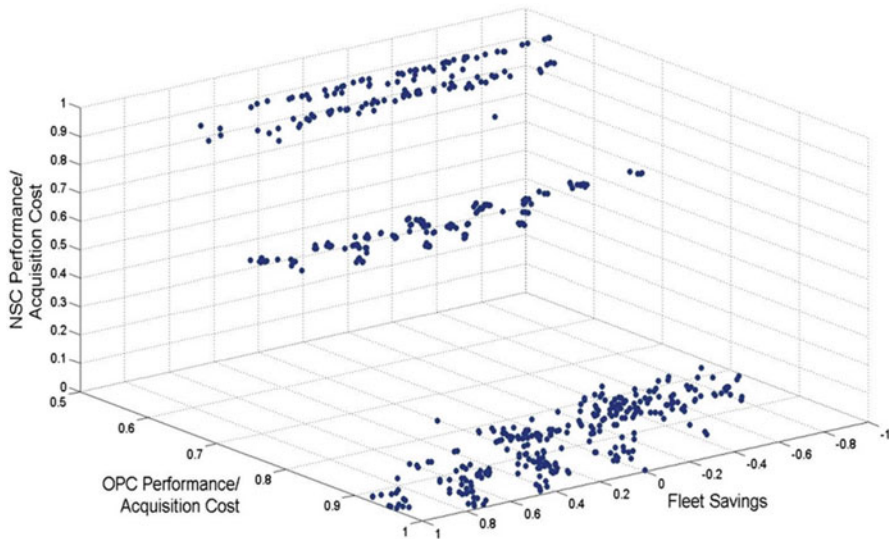


Fig. 25.11 Optimization results for a 288 possible commonality combinations

When the 159 positive net fleet savings commonality pairs were sorted to determine those nondominated designs that lie on the discrete Pareto surface, only 20 different commonality combinations remained. Of these there were only 12 uniquely different design pairs from a naval architectural standpoint. These are as shown in Fig. 25.12 with the baseline, no commonality designs that yield no net fleet savings (best NSC and best OPC). The design pair NSC<sub>15</sub> and OPC<sub>15</sub> (using the 46 mm gun, the smaller cruise engines, the smallest ship service diesel generator sets, the small superstructure, and the small midship section blocks in common) yields the greatest overall fleet savings from their commonality. Note, however, that the NSC<sub>15</sub> design has a significant performance loss compared to the baseline design, primarily from its use of only one helicopter hangar. The NSC<sub>18</sub> and OPC<sub>18</sub> designs (using the smallest ship service diesel generator sets and the large superstructure in common) have the largest net fleet savings before the shift from two helicopter hangars to one. Thus, they are at a “knee” of the surface and are particularly attractive designs.

The characteristics of the NSC<sub>18</sub>/OPC<sub>18</sub> pair and the NSC<sub>15</sub>/OPC<sub>15</sub> pair of designs are summarized in Table 25.3. Note that the NSC<sub>15</sub> design has a significant (52.4 %) performance loss compared to the baseline design, primarily due to its use of only one helicopter hangar. Note also that the two 15 designs are probably close enough that it might be better to produce one design for both missions and save even more by complete commonality. Note that the NSC<sub>18</sub>/OPC<sub>18</sub> pair provides 97 % of the NSC baseline performance, 100 % of the OPC baseline performance, and still provide 60.7 % of the maximum net fleet savings observed.



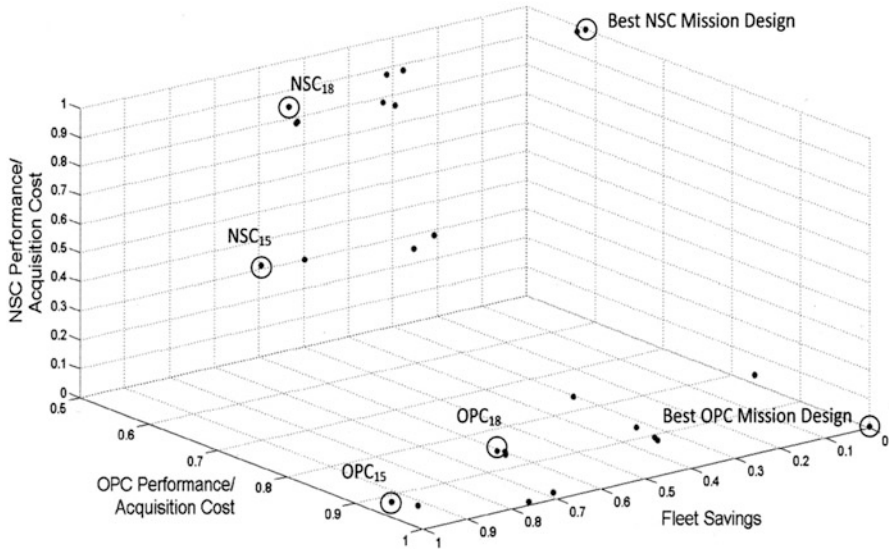


Fig. 25.12 Three-objective discrete Pareto surface

Table 25.3 Characteristics for selected Pareto front designs (adapted from Corl et al. 2007b)

Point	L m (ft)	B m (ft)	V <sub>max</sub> kts	kW <sub>max</sub> (SHP)	V <sub>cruise</sub> kts	kW <sub>cruise</sub> (SHP)	Range nm	OPC Perf.	NSC Perf.	Cost \$mil	Fleet Savings \$mil
OPC <sub>18</sub>	107.6 (353)	16.46 (54)	22.0	5,757 (7,720)	18.0	2,895 (3,882)	9,158	100.0	0.314	89.8	45.5
NSC <sub>18</sub>	121.6 (399)	16.46 (54)	27.9	16,687 (22,377)	18.0	3,585 (4,807)	12,074	100.0	97.0	141.7	45.5
OPC <sub>15</sub>	91.4 (300)	12.19 (40)	22.2	5,333 (7,152)	18.0	2,537 (3,402)	9,046	89.7	2.95	72.9	75.0
NSC <sub>15</sub>	92.4 (303)	12.19 (40)	25.5	9,642 (12,930)	18.0	2,767 (3,710)	9,019	89.7	47.6	91.1	75.0

## 25.6 Conclusions

The methodology contained in this chapter offers a valuable tool for use in making optimal commonality decisions for small-volume applications. It provides a logical procedure for the use of commonality in design while taking into consideration performance loss, cost, and savings. In much of the commonality literature, there is a basic assumption that commonality in design hinders the performance of products, and this is accepted because of the savings associated with using common parts. It is simply assumed that the use of commonality always results in savings. Prior to this methodology, the amount of savings was never explicitly quantified. The sample results in this chapter show that positive savings is not always realized.

If poor commonality decisions are made in design, products could both cost more and perform less.

The mission performance model presented relies on the use of fuzzy utility values. Performance was determined using the four design characteristics for each mission area and applying the corresponding fuzzy utility value to each. Studies were performed which demonstrated the method's sensitivity to changing these fuzzy utility values. A designer could easily modify this model to include more design characteristics or even more mission areas. The fuzzy utilities could also be replaced with another tool for awarding value to a given design characteristic.

In this chapter, commonality decisions were limited to five components. Each of these components was integrated into the design in a slightly different manner to show the versatility of the optimization model. In the sample presented there were a finite number of commonality options from which to choose. As a result, an exhaustive search was used to determine which commonality choices were the best. In this case, the exhaustive search saved computation time. However, if more choices are available, another evolutionary algorithm could be used to more efficiently search for the Pareto optimal commonality combinations.

Bulk purchasing and construction learning curves were used in the example to determine the savings associated with the use of commonality. The savings model was kept relatively simple. Other forms of savings could be realized as well. These could include training of personnel, technical design costs, administrative savings, facility costs, and spare parts. The type of savings and the number of different factors to consider varies with each product being designed. A designer may choose to make the savings model very elaborate when information of these other forms of savings is available, or it may be kept simple as seen in this chapter.

Using the logical methodology presented here will enable a designer to present a much more complete analysis of commonality decisions in design. Designers can expand the optimization model in many ways to adapt it to their particular needs. Regardless of how crude or elaborate it may become, the process shown should be able to stay very much the same.

**Acknowledgments** This development of the commonality optimization methodology described here was possible with academic leave and related support of the US Coast Guard for the lead author and support from the Office of Naval Research through N00014-03-0983 for all authors. The authors would like to thank the US Coast Guard for the use of their ship synthesis model and emphasize that this research is by no means intended to be a comparison to any US Coast Guard design work for either the OPC or the NSC. The initial mission requirements for the endurance cutter designs were used for academic purposes only.

## References

- ASSET (2005) Advanced surface ship evaluation tool. Naval Surface Warfare Center, Carderock Division, Version 5.2.0
- Ayyub BM (2002) Elicitation of expert opinions for uncertainty and risks. CRC, Boca Raton, FL

- Brown A, Salcedo J (2003) Multi-objective optimization in naval ship design. *Nav Eng J* 115:49–61
- Corl MJ (2007) Methodology for optimizing commonality decisions in multiple classes of ships. Dissertation, University of Michigan
- Corl MJ, Kokkolaras M, Parsons MG (2007a) Platform-based design of a family of ships considering both performance and savings. In: Proceedings of international conference on engineering design, ICED'07, Paris, France
- Corl MJ, Parsons MG, Kokkolaras M (2007b) Methodology for the optimization of commonality in multiple ship classes. *Trans Soc Nav Archit Mar Eng (SNAME)* 115:68–93
- Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York
- Fellini R, Kokkolaras M, Michelena N, Papalambros PY, Saitou K, Perez-Duarte A, Feynes P (2004) A sensitivity-based commonality strategy for family products of mild variation with application to automotive body structures. *Struct Multidiscip Optim* 27:89–96
- Fellini R, Kokkolaras M, Papalambros PY, Perez-Duarte A (2005) Platform selection under performance loss constraints in optimal design of product families. *J Mech Des* 127:524–535
- Fellini R, Kokkolaras M, Papalambros PY (2006) Quantitative platform selection in optimal design of product families, with application to automotive engine design. *J Eng Des* 17:429–446
- Fujita K, Yoshida H (2004) Product variety optimization: simultaneously designing module combination and module attributes. *Concur Eng Res Appl* 12:105–118
- Goldberg DE (1989) Genetic algorithms for search, optimization, and machine learning. Addison-Wesley, Reading, MA
- Gonzalez-Zugasti JP, Otto KN (2000) Modular platform-based product design. In: Proceedings of the 2000 ASME design engineering technical conference, Baltimore, MD
- Gonzalez-Zugasti JP, Otto KN, Baker JD (2000) A method for architecting product platforms. *Res Eng Des* 12:61–72
- Kosko B (1992) Neural networks and fuzzy systems: a dynamical systems approach to machine learning. Prentice-Hall, Englewood Cliffs, NJ
- Li J, Parsons MG (1998) An improved method for shipbuilding market modeling and forecasting. *Trans Soc Nav Archit Mar Eng* 106:157–183
- Messac A, Martinez MP, Simpson TW (2002) Effective product family design using physical programming. *Eng Optim* 34:245–261
- Michalewicz Z (1996) Genetic algorithms + data structures = evolutionary programs. Springer, Berlin
- Naval Surface Warfare Center Carderock Division (1998) User's guide USCG performance based cost model
- Nayak RW, Chen W, Simpson TW (2002) Variation-based methodology for product family design. *J Eng Optim* 34:65–81
- Oszycza A (2002) Evolutionary algorithms for single and multicriteria design optimization. Physica, Berlin
- Parsons MG (2003) Parametric design. In: Lamb T (ed) Ship design and construction. SNAME, Jersey City, NJ
- Parsons MG (2009) Applications of optimization in early ship design. *Ship Sci Technol (Rev Cienc Technol Buques)* 5:9–31
- Saaty TL (1996) The analytical hierarchy process. RWS, Pittsburgh, PA
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf* 18:3–20
- Simpson TW, Maier JRA, Mistree F (2001) Product platform design: method and application. *Res Eng Des* 13:2–22
- Simpson TW, Siddique Z, Jiao J (eds) (2005) Product platform and product family design: methods and application. Springer, Berlin
- U.S. Coast Guard Internal Documents, various

- U.S. Coast Guard Memorandum (1995) Mission analysis report (MAR) N-001-95 deepwater missions
- U.S. Coast Guard (2006) <http://www.uscg.mil/deepwater/system/cuttercomparison>. Accessed 15 Dec 2006
- Zalek SF (2007) Multicriterion evolutionary optimization of ship hull forms for propulsion and seakeeping. Dissertation, University of Michigan
- Zalek SF, Parsons MG, Papalambros PY (2006) Multi-criteria design optimization of monohull vessels for propulsion and seakeeping. In: Proceedings of the 9th international marine design conference, vol 2, pp 533–557
- Zitzler E, Laumanns M, Bleuler S (2003) A tutorial on evolutionary multi-objective optimization. <http://citeseer.ist.psu.edu/zitzler03tutorial>. Accessed 4 June 2003

# Chapter 26

## A Heuristic Approach to Architectural Design of Software-Intensive Product Platforms

Carlos O. Morales

**Abstract** This chapter introduces a heuristic approach for the analysis, architecting, and design of software-centric product platforms. The central role of software architecture is stressed by highlighting its relationship to the analysis of new product domains. Several case studies are used to illustrate key concepts, including a more detailed case on the design of an object-oriented application framework as platform for a family of products that control industrial processing machines. Case studies and methodology are linked to important software engineering design principles. At the end of the detailed case study, an approximate measure of code reuse and its economic impact is presented, which can serve to support the business case of making the significant investment required by a software platform for a family of related products. This chapter builds on fundamental software engineering concepts introduced in Chap. 21.

### 26.1 Introduction

Each new generation of high-technology products is smarter and more sophisticated than the previous one, mainly as a result of their advanced software features. Voice recognition and synthesis, automated suggestions for new purchases, smart assistants that anticipate the user's intention, devices that learn the user's preferences and lifestyle, expert systems that can perform automated diagnoses and classifications, or predictions of the future cost of airline tickets and stock prices. All of these technologies are manifestations of the ever-increasing complexity of software.

Software is inherently complex, and its malleability coupled with the sheer number of degrees of freedom (DOF) of a software system (one DOF per line of code) makes it a fragile and fertile source of product defects. However, as software

---

C.O. Morales (✉)  
Animas Corporation, Johnson & Johnson Medical Devices, 200 Lawrence Drive,  
West Chester, PA 19380, USA  
e-mail: [carlos.o.morales@ieee.org](mailto:carlos.o.morales@ieee.org)

engineering continues to mature as a discipline, new methodologies, tools, processes, and design approaches aid in the development of more solid and robust software technologies for this kind of products.

Mission-critical systems, in particular, require a high level of discipline and formality during their design, implementation, and testing. It is of the utmost importance to spend enough time analyzing and understanding the specific domain for which the new product is targeted. Examples of mission-critical systems include implantable medical devices, life-support equipment in hospitals, weapon systems, avionics, spacecraft, telecommunication satellites, nuclear reactor controllers, and automobile control computers, just to name a few. Nevertheless, the same process and concepts can also be applied to other products like cellular phones, industrial controllers, or consumer video equipment, where overall quality of the product is increasingly judged by the quality of its software content.

These sophisticated products must be built on a solid foundation if they are to perform safely and effectively, and the best way to accomplish this is to design and implement a software platform for a family of related products, which is known in software engineering as a domain-specific framework (Fayad and Schmidt 1997). One of the most notable qualities of these frameworks is that their robustness is improved every time this platform is reused to derive a new product, since the framework constitutes a reusable reference design and a very valuable, thoroughly tested, reusable code base.

Domain-specific application frameworks, or enterprise frameworks, are neither easy nor cheap to develop. The task requires a dedicated team of professional software engineers led by an architect with a deep understanding of the problem domain and with the experience of having previously developed several applications in that same domain. For more information on enterprise frameworks, their challenges, and economic justifications, the reader is referred to CACM (1997).

In this chapter, we use examples from several domains to illustrate the analysis and design of software systems that could serve as generalized solutions, or platforms, for families of related products.

## 26.2 Definitions of Framework in Software Engineering

The term “framework” is heavily used in software engineering, but it can be confusing sometimes. For clarity, we shall define the following terms: “enterprise architecture framework,” “software infrastructure framework,” and “family platform framework.”

### 26.2.1 *Industry-Standard Enterprise Architecture Framework*

IEEE Standard 42010–2011, “Systems and Software Engineering: Architecture Description” (IEEE 2011) is now an international standard that has also been adopted by ISO and IEC. This document specifies *the manner in which architecture*

*descriptions of systems are organized and expressed.* This includes specifications for *architecture viewpoints, architecture frameworks, and architecture description languages for use in architecture descriptions.* This standard defines the term “architecture framework” as *conventions, principles, and practices for the description of architectures established within a specific domain of application and/or community of stakeholders.*

One of the earliest publications on the topic of enterprise architecture frameworks was authored by John Zachman, of IBM (Zachman 1987). The Zachman Framework continues to be in use today, and it paved the way for the creation of architecture frameworks for many domains, most notably for the defense industry, e.g., DoDAF, the US Department of Defense Architecture Framework standard on how to document architectures. Further discussion on architecture frameworks is beyond the scope of this chapter, and the interested reader is referred to Clemens et al. (2011) for an accessible overview of DoDAF and other architecture frameworks. For a survey of all known architecture frameworks currently in use, please see ISO (2011).

The main idea to remember here is that industry-standard architecture frameworks are not executable code and refer mainly to documentation requirements.

### ***26.2.2 Software Infrastructure Framework***

Examples of these frameworks include Microsoft Foundation Classes (MFC<sup>®</sup>), Microsoft NET Framework<sup>®</sup>, Java EE<sup>®</sup>, and frameworks for creating graphical user interfaces (GUI toolkits) such as Qt, Motif, Swing<sup>®</sup>, or Adobe Flash<sup>®</sup>, just to name a few. This class of frameworks has been called by many names, including *application frameworks and architectural frameworks.* These frameworks are collections of software libraries that provide infrastructure services for other applications to use in the form of software objects. Some notable classes provided by these frameworks are GUI widgets, networking services, web services, e-mail, and other messaging services.

These frameworks provide executable code that is intended for general purpose, and they can be used to build sophisticated software applications by using its high-level functionality. However, they do not solve problems that pertain to a particular domain or business, which is the next level up in the software abstraction scale, as described below. A software infrastructure framework can be thought of as a general-purpose toolbox.

### ***26.2.3 Family Platform Framework***

In this work, we refer to enterprise application framework as a former name for family platform framework, since the term helps to make the connection with the literature. However, we prefer to use the term family platform framework because it is more accurate and descriptive regarding its intention and use.

Enterprise application frameworks, or family platform frameworks, implement solutions for a specific domain. This kind of framework is a reusable, semi-complete application that can be specialized to produce custom applications. They are designed for particular businesses such as data processing, telecommunications, or other industrial domains. A platform framework reuses not only code but design as well. It describes how the system is decomposed into cooperating objects and how custom applications must be implemented based on this infrastructure. Platform frameworks are the software foundation for families of related products, i.e., they serve as software product family platforms.

A good introduction to enterprise application frameworks can be found in the literature (CACM 1997). For a deeper study of all that entails to embark in their design and development, see the referenced works by Fayad et al. (1997, 1999, 2000), Schmidt and Fayad (1997), Schmidt et al. (2000), and Johnson (1997).

The term “family platform framework” specifically refers to an object-oriented enterprise application framework that serves as the foundation for a family of related products and that has been developed according to the specific software architecture and design process described herein.

Later, a detailed case study is introduced to illustrate the design and implementation of a software platform for a family of industrial machines and the economic impact of creating software family platforms. The scope of this chapter is to present the thought process for analyzing a newly planned product family and for mapping the results of that analysis to the family architecture and its detailed software design.

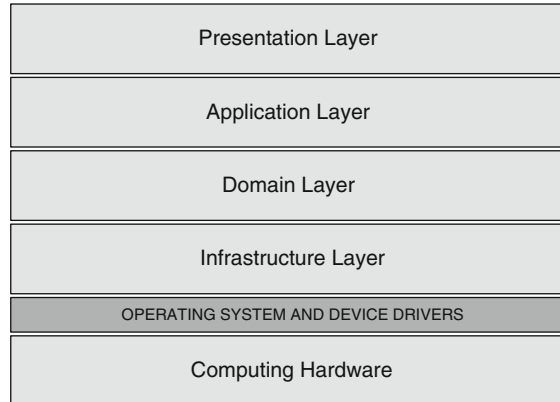
## 26.3 General Architecture of Software-Intensive Products

This chapter uses, and builds on, the concepts presented in Chap. 21 on software design principles (Morales 2013). It is recommended to read that chapter first, if the reader is unfamiliar with some of the terms used below.

Most modern software-intensive products are embedded devices, but not necessarily. As shown in the following examples, these products can be found hidden in vehicles, in the form of high-volume consumer devices, high-cost customizable industrial equipment, or just as pure software residing on remote servers providing service to clients over the Internet (also known as “the cloud”).

The best way to tame the complexity of most software-intensive modern products is with the application of the age-old principle of *Divide and Conquer*. Figure 26.1 shows a typical layered architecture for software applications, based on the *LAYERS* architectural pattern (Buschmann et al. 1996), which is widely used in industry and even standardized for some applications like networking. The main advantage of this architectural pattern is that each layer specializes in a particular aspect of the application. Each layer is highly cohesive and is loosely coupled with its adjacent layers. Typically, dependency among layers only flows in one direction,



**Fig. 26.1** Layered architecture

i.e., downwards. This means that upper layers can typically initiate interaction with the lower layer at any time by requesting services through its interface.

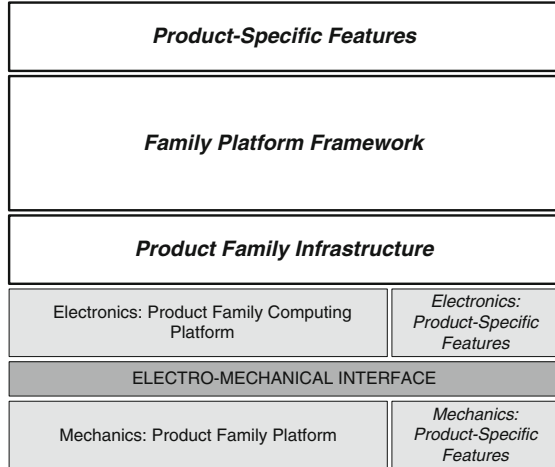
In the classical structure of a layered architecture, the Infrastructure software layer provides generic technical services that enable access to the system's resources, while the Domain layer encapsulates the business-specific concepts and rules. The Application layer implements particular jobs that the software is intended to do, but it does not include any business knowledge. It accomplishes its task by delegating processing and coordinating cooperation between objects in the Domain layer. The Presentation layer is typically the user interface.

Let us use an example to illustrate how this structure is applied. In an online banking application, the Presentation layer would be implemented within the customer's web browser, through which she navigates the system's functionality and requests transactions. The Infrastructure layer would provide user authentication services, access to the bank's databases, and encryption for a secure connection between the bank and the customer, among many other facilities.

In the case of a fund transfer operation, for instance, the customer would enter information through the Presentation layer (GUI), which in turn would send it down to the Application layer. The Application layer is responsible for validating that input and then sending it down to the Domain layer for processing. Data sent would include the amount to be transferred, currency type, source account number, and destination account number, and that is the end of its responsibility. The Domain layer implements the essential business rules for the banking business, like accounting, for example. In accounting, one of the fundamental business rules is "Every credit must have a matching debit"; thus, it performs the operation by modifying all the necessary tables in the bank's database through requests to the Infrastructure layer, which provides these services. As each operation in the lower layers is completed, the upper layers are notified, until the end result is finally presented back to the customer at the top layer.

When we embark on the task of designing a platform for a product family, maximizing software reuse is one of the main objectives. Although designing and

**Fig. 26.2** General architecture for a software-intensive product platform



implementing a platform take time, it later pays off significantly. Maximizing code reuse ensures that once the platform is finished, new products can be implemented in a very short time, and it allows engineers to focus on what makes the new product unique, i.e., they don't have to reinvent everything with each new project.

In order to maximize code reuse, we design and implement an object-oriented application framework to work as the Domain layer, which here we call the *family platform framework*. We call it by a different name because it not only implements the core functionality that comprises the essence of the product family as well as its overall design philosophy but, at the same time, it imposes certain rules for the implementation of features that differentiate individual members of the family, both in software at the top layer and in hardware at the bottom layer. At the top, we take the layered architecture described above and compress the Application and Presentation layers into one, which we call the *product-specific features*. Individual product-specific features could be optionally included or excluded in order to create different products within the family, even though their structure and basic behavior are dictated by the platform framework architecture.

The advantages of merging the two top layers into one will become clear with some examples, as described later, but the main reason is that the framework takes control over most things in the product family, and user interfaces (Presentation) become just one more product-specific feature that must comply with the interfaces prescribed by the platform.

At the bottom of the software layer hierarchy we have the *Product Family Infrastructure* layer, which is slightly different from the classical banking example presented above. In the case of a product family platform, the Infrastructure layer must not be static, nor monolithic, but modular and extensible to accommodate the evolving hardware needs of individual products within the family. This is illustrated with the partitioned hardware layers shown in Fig. 26.2, where we can appreciate a standardized hardware platform for the family, plus additional modules that are product specific.

This highly cohesive domain encapsulation centered on the family platform framework brings along a technical risk. If the platform is not designed correctly, then the number of products that can be easily derived from the platform would not be as prolific as expected. Investment on a product platform is justified only when it has a long life span, consistently generating a long sequence of derived products. Two key principles that will help us achieve a correct design of the platform are *Abstraction* and *Design for Change* (Morales 2013).

Figure 26.2 shows a generalized layered architecture for a software-intensive product platform, which is based on the same concepts of the *LAYERS* architectural pattern, but tailored to suit the specific needs of a software platform for a family of related products. This figure describes a platform for embedded devices more accurately, although it also applies to software-only products if we remove the bottom layer representing the product's mechanisms. The key concept here is *abstraction*, clearly *separating concerns* into specific containers that segregate technologies and encapsulate those features that belong to the common product platform and those that distinguish individual products in the family.

Each layer is abstracted away from the adjacent layers by means of their interfaces. The framework exposes an Application Programming Interface (API), which prescribes the behavior and data expected from the software modules that comprise the Product-Specific Features' layer. The electromechanical interface specifies how the electronics and mechanical subsystems fit within the system in order to interact with the external world. Compared with the reference *LAYERS* architectural pattern, the operating system and device drivers' layer, which isolates the software from its computing platform, have been merged into the family platform framework.

In order to design an effective and efficient product family platform, we must make design choices related to the computing platform that will ensure software compatibility across all derived products in the family and maximize reuse. Specifically, the platform specification should prescribe a particular processor core for the whole family, e.g., an ARM Cortex or an Intel Atom, or some other specific processor. However, this choice does not prevent product designers from having a rich assortment of peripherals around the processor core that can be very different from product to product within the same family—as long as they keep the same core. Furthermore, choosing a single operating system for the whole family makes software reuse even easier and more cost-effective.

A successful family platform framework typically implements approximately 80 % of the functionality for each product derived from the family, and it prescribes with precision how the remaining 20 % (product-specific features) are to be designed and implemented. Although it sounds restrictive, it is in fact very effective and efficient, since the enforcement of the family architecture and its behavior encapsulated in the platform forces and enables software engineers to follow a clear and consistent process for the implementation of derivative products based on the platform. This phenomenon will be explained in more detail later in this chapter.

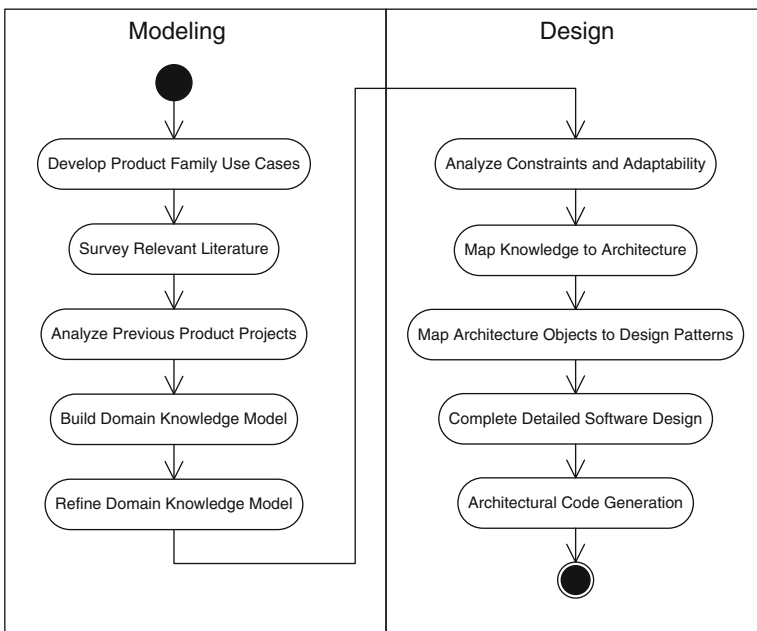
## 26.4 Domain Analysis

Deep knowledge about a product domain doesn't come easy. The best situation is to have previous experience designing stand-alone products for that domain before one commits to designing a successful product family platform, or at least, to have a multidisciplinary team of product designers and consultants with significant experience in that particular domain. The reason for this concern is related to the technical risk mentioned above: if the product family framework is not designed correctly, then the exercise will result in a significant expense that does not meet expectations. This is not a typical software project and has many subtleties in designing and building object-oriented enterprise application frameworks (Fayad and Johnson 1999; Fayad and Schmidt 1997).

Figure 26.3 shows a UML activity diagram that describes the high-level process for synthesizing the core essence of a product domain and incorporating that knowledge into a new product family platform. Each step is described below.

### 26.4.1 *Develop Product Platform Use Cases*

When designing a platform for a new product family, the first thing we need to know is a clear definition of its scope. It is necessary to specify the behavior and functionality that these new products must exhibit in order to satisfy the needs of all stakeholders.



**Fig. 26.3** High-level process for synthesizing a product family platform

One of the best ways to obtain this understanding is to analyze the expected use of the product family through use cases (Jacobson et al. 1992).

The scope of the use cases must remain at a high level of abstraction, where the goal is to obtain all the scenarios in which each type of stakeholder will interact with a generic product of this new family in order to satisfy their goals. Some specific needs may be satisfied by one product derived from the platform, but at this point, the analysis should be kept at the product platform level. Our goal at this stage is to abstract the essential use cases that make up the core behavior of the product family. An excellent manual for analyzing use cases is Cockburn (2000).

### ***26.4.2 Survey Relevant Literature***

This activity is particularly important (and indispensable) for software designers that are newcomers to the particular domain of interest. Collect all relevant literature about the domain, which includes books, journal publications, presentations, technical descriptions, and specification documents for similar products that can be used as precedents. Reliability of all sources should be confirmed, ensuring that they represent the consensus of subject matter experts. Extract and abstract the essential knowledge about the domain, summarizing the essential concepts in tables or some other tool.

### ***26.4.3 Analyze Previous Product Projects***

Fred Brooks said that, when designing a new kind of system, software teams *will* throw one system away whether they want it or not (Brooks 1975). His argument was in favor of using pilot projects to ensure that we can ultimately deliver exactly the software system that we all want. There is certainly great value in having the opportunity to perform forensic analysis of previous projects that have attempted to solve the same problem that we are now facing, since this improves the odds that the team will arrive at a better design the next time. However, our focus here is not on the software design or the code. What we are looking for are abstract concepts: those ideas that represent the essence of the kind of products for which we want to design a platform and those characteristics, functions, and behavior that comprise the essence of being a member of that family of products. For example, we want to discover those essential attributes and behavior that constitutes being a smartphone, or a pacemaker, or a human-size robot arm.

Essentially, this is still part of the information survey that began with the relevant literature review, but now our sources are more heterogeneous and focused on the actual product needs. We have to continue extracting and abstracting important knowledge about the domain from these new sources and add the most relevant concepts and relations to the compilation we started with the literature review.

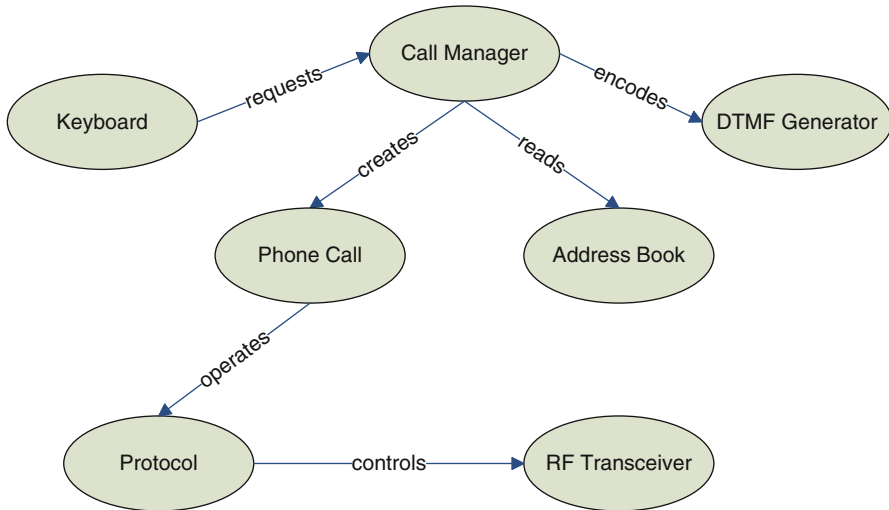


Fig. 26.4 Knowledge graph partially representing the mobile phone domain

#### 26.4.4 Build Top-Level Domain Knowledge Model

Using the knowledge derived from the literature and product information surveys, we now build a knowledge graph (Fayad et al. 1999). Knowledge graphs are a form of the more widely known Semantic Networks (Russell and Norvig 2009).

A knowledge graph is made up of nodes and edges. Nodes are those concepts that appear consistently and in similar ways, in the domain knowledge survey. These essential domain concepts are typically nouns, and each node represents an indispensable concept for the product platform. Edges represent relations among the different concepts, or nodes, and they are typically represented with verbs indicating actions that one node performs on other nodes.

As an example, Fig. 26.4 shows a knowledge graph that abstracts and represents some of the concepts and relations that could be used to design a generic product in the mobile phone domain. Notice that the nodes represent concepts that are indispensable for the product to be considered a mobile phone. In other words, they constitute the essence of the product family.

The key design principles here are *Separation of Concerns*, *Abstraction*, and *Generality* (Morales 2013).

#### 26.4.5 Refine Domain Knowledge Model

Once we have collected the main concepts and relationships in the top-level domain model, it is necessary to investigate the internal structure of each node in order to gain a deeper understanding of the platform design needs. This refinement of the

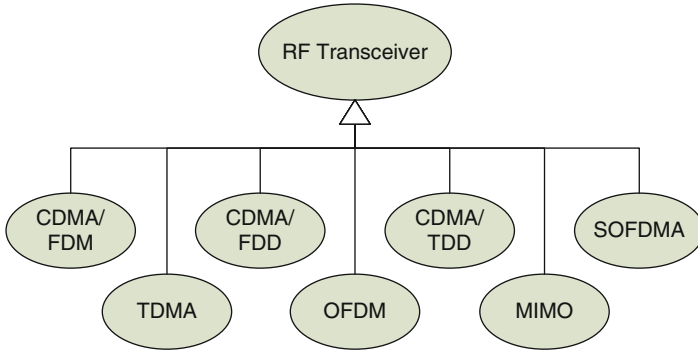


Fig. 26.5 Refined knowledge graph for the RF Transceiver node in the mobile phone domain

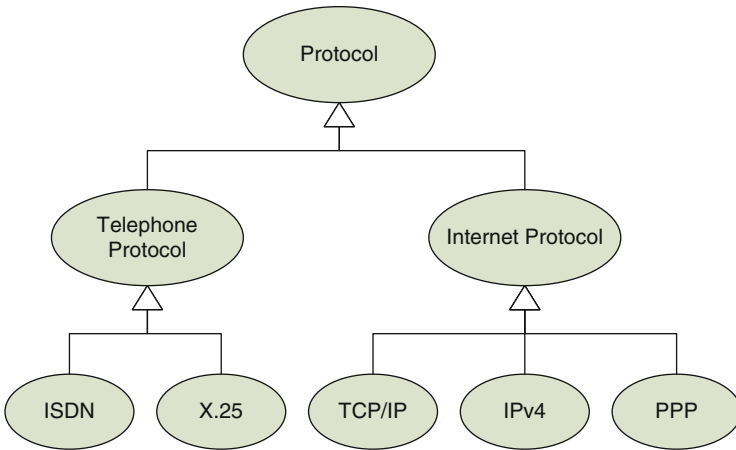


Fig. 26.6 Refined knowledge graph for the Protocol node in the mobile phone domain

domain knowledge model reveals more information that is less abstract and more suitable for use in the design of the product platform.

Figures 26.5 and 26.6 show the concept hierarchies for the RF Transceiver and Protocol nodes. The white triangle denotes specialization of the root concept, similar to inheritance in object-oriented software. The root node can be thought of as similar to an abstract class and the leaves as similar to concrete derived classes.

### 26.4.6 Analyze Constraints and Adaptability

This is a critical step in the process of abstracting knowledge about a domain and using that knowledge to design a successful product family platform that can evolve over time and be useful to generate a prolific family of derived products.

The key design principle here is *Design for Change* and its various forms of expression: *Change of Algorithms*, *Change of Data Representation*, *Change of Abstract Machine*, *Change of Peripheral Devices*, and *Change of Social Environment* (Morales 2013).

Each node in the knowledge graph must be analyzed for its potential need of change in the future due to foreseeable technological progress or to enable different features for distinct members of the product family. All anticipated changes can be compiled in tables, supported by reference documents.

For example, let us continue using the mobile phone example, and oversimplifying for the sake of clarity, we say that it could be anticipated that the “RF Transceiver” node will change. This node represents the radio transmitter and receiver used by the mobile phone to connect to Base Stations, and it could use any of two multiplexing technologies, namely, code division multiple access (CDMA) or time division multiple access (TDMA), depending on the target market of the particular product. This is an indicator of the need for *Modularity* in the implementation of this feature in the product platform. Compliance with each of these standard technologies is also a constraint on the system.

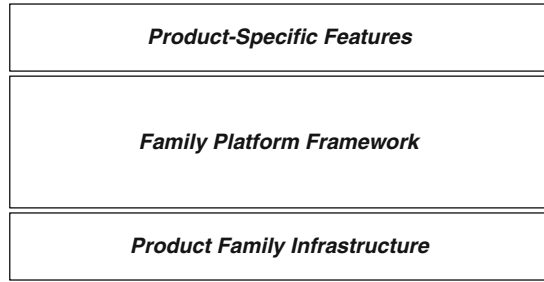
Similarly, and oversimplifying again, we could see that the “Network Protocol” node should be easily adaptable to work with potentially non-compatible new technologies. Experience shows that mobile network technology evolves very quickly, and thus, for the investment on a long-life span product platform to be worthwhile, it should be able to adapt to these new technologies as they come along. In this manner, we have seen three generations of mobile networks based on the Global System for Mobile Communications (GSM), including general packet radio service (GPRS) for second-generation networks (2G), Universal Mobile Telecommunications System (UMTS) for third-generation networks (3G), or Long-Term Evolution for fourth-generation networks (4G LTE). These are examples of *Constraints* on the platform design and *Change of Peripheral Devices*.

Other concepts are not as clear-cut, however. For example, a keyboard could have different embodiments in two products derived from the same platform. Let’s say that one product could use a membrane keyboard driving the electronics directly, and another could implement it as a soft keyboard on a touch screen, driving a software device driver instead (*Change of Peripherals* or *Change of Data Representation*). However, at the product platform level, the abstract concept of a keyboard is exactly the same. Therefore, the actual implementation of the keyboard in a particular product is a secondary matter. Along the same lines, the abstract concept of an address book is exactly the same at the platform level, even though it might be implemented as a remote web page that is accessed over the Internet or as a local database file, which, as described above, would constitute secondary concepts and not essential details for the platform.

The dual-tone multi-frequency (DTMF) generator can be implemented in one way only, since it is a real function based on an international industry standard with which every phone, regardless of its technology, must comply. This is a *Constraint* on the system design. In contrast, the concept of a “Call” is completely abstract, completely defined internally, and its representation could be unique to the product platform.



**Fig. 26.7** Layered software architecture for a product family platform



Although the user interface is not shown in the knowledge graph, we can say that the product platform software should be designed to represent all the concepts of a user interface in an abstract form within the platform, e.g., using label IDs instead of actual text strings, and anticipate a *Change of Social Environment* by designing the platform in such a way that different languages can be easily implemented through loosely coupled software components that are external to the platform.

#### **26.4.7 Map Knowledge to Architecture**

With a greater understanding of each node and its internal structure in the domain knowledge graph, we now proceed to find the right place for each node in the software architecture. Figure 26.7 shows the software section of the generalized architecture presented earlier. Assignment of each node to the appropriate software layer will ensure the construction of layers that have the very important properties of *high cohesion* within them and *low coupling* between them (Morales 2013).

In general, the Product Family Infrastructure layer should provide access to the physical resources of the system, the Family Platform Framework layer should encapsulate all the system behavior and functionality that does not change from product to product, and the top layer should encapsulate those features that distinguish each product from the rest of the family.

In the mobile phone example, it is clear that abstract keyboard and RF Transceiver refer to physical resources and, therefore, belong in the Infrastructure layer. The call manager, phone call objects, Internet session objects, protocol handlers, address book, and DTMF generator all belong specifically to the mobile phone domain, i.e., in the family platform framework. Note that none of the nodes in the top-level knowledge graph is actually mapped to the Product-Specific Features layer directly, but as revealed by the Constraints and Adaptability analysis above, some of the nodes have deeper hierarchies where some of the sub-nodes are expected to change over time or change from product to product in the family. In other words, some of the sub-nodes should be allocated to the top layer where they

function as product differentiators. Those sub-nodes that are not expected to change, regardless of the particular product instance, should be allocated to the family platform framework to be encapsulated. The best way to produce a robust arrangement of software objects is to use a catalog of best design practices and time-proven solutions known as design patterns (Gamma et al. 1995).

### ***26.4.8 Map Architecture Objects to Design Patterns***

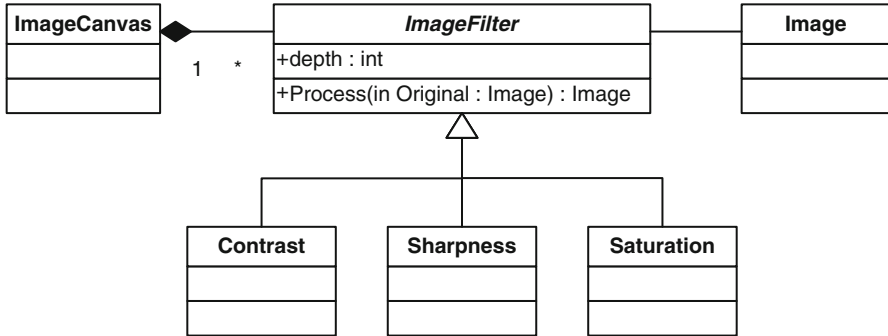
The Constraints and Adaptability analysis guides the selection of the best design pattern for each case. For example, some of the classic design patterns from Gamma et al. (1995) are the following: “Bridge,” which decouples an abstraction from its implementation so that both can vary independently; “Proxy,” which provides a surrogate for another object in order to control access to it; and “Strategy,” which defines the interface for a family of algorithms, encapsulating them and making them interchangeable. After the landmark work of Gamma, Helm, Johnson, and Vlissides, many other books on design patterns have been published. Some of them are new catalogs of design solutions for software in general, and some others are aimed at particular domains (e.g., Johnson 1992; Buschmann et al. 1996; Schmidt et al. 2000; Fowler 2002; Douglass 2002, 2011; Alur et al. 2003; Daigneau 2011).

As an example, let us assume that our family of smartphones has a built-in image enhancement tool designed to improve the quality of images taken with the phone’s built-in camera. A digital image is represented in software as a matrix of pixels with color values. Users can perform automatic or manual enhancements on a single image by running image filters to enhance sharpness, contrast, modify color saturation, or exposure (brightness). Since this is a feature that all products in the family will have, these image-enhancing mechanisms will be allocated to the Family Platform Framework layer. All image filters operate as convolutions on the original image and produce a new image as output, with modified values on each pixel according to the requested operation. Figure 26.8 shows a UML class diagram that implements a family of image processing algorithms according to the “Strategy” design pattern (Gamma et al. 1995).

### ***26.4.9 Complete Detailed Software Model Design***

Applying design patterns is only the beginning of detailed software design. Design patterns are coarse-grain building blocks, but each software system has nuances and peculiar concepts that need to be represented in detail, as well as the interconnection among design pattern structures.

The ever-increasing complexity of each new generation of high-technology products requires the use of appropriate tools to handle it and enable designers to



**Fig. 26.8** An image processing feature for a family of mobile smartphones based on the Strategy design pattern

look at the system from different perspectives. Abstraction is the strongest feature of model-based development, also known as Model-Driven Architecture (MDA). By slicing the system into distinct perspectives that show its structure, dynamic behavior, interfaces, and internal states, designers can develop a complex system that maintains consistency and correctness throughout the product development life cycle. The Unified Modeling Language, or UML, was specifically developed for this purpose. For more detailed presentations, the reader is referred to the following: Fowler (2003), Booch et al. (1998), Rumbaugh et al. (1998), and OMG UML (2011).

### 26.4.10 Architectural Code Generation

An additional benefit of modern software modeling tools based on UML is that most of them provide code generation facilities, as well as two-way code engineering. We use the term architectural code generation to refer to the automatic generation of source code in an industry-standard programming language like C++, Java, C#, or any other. This automatically generated code typically includes all software interfaces defined as properties and methods of a class, as well as processor and project-specific header files. Since the Object Management Group (OMG) released the new specification for UML 2.0, executable models are now possible, and code generated automatically also includes source code that implements state machine behavior directly from UML State Diagrams (also known as Statecharts).

Two-way code engineering tools allow software developers to maintain the software model and source code synchronized by having the tool automatically update the code every time the model is changed. Likewise, it can also update the model whenever the source code is modified, e.g., when a function parameter changes name or data type.

## **26.5 Case Study: Software Platform for a Family of Industrial Machines**

### **26.5.1 System Overview**

We now present a more detailed case study from an actual project to illustrate the design of a software platform for a product family. Here, we review the characteristics of the system and the design decisions that made it a successful platform. The purpose of this system was to serve as a generalized solution that addresses the specific requirements of a family of products in the industrial automation and test domain.

The main purpose of this platform is to provide the software foundation for a family of automated machines that are used in industrial manufacturing and test. Typical applications of this product family are stand-alone units, or cells, that are integrated into fully automated or semiautomated manufacturing lines. Many of these cells include one or more robots for product handling, machine vision systems for robot guidance and automated visual inspection, general-purpose digital and analog inputs and outputs and programmable power supplies, waveform generators, current and voltage meters, and other instrumentation equipment. The same robots, digital cameras, power supplies, and meters can be used across many different projects, for example, assembling handheld blood glucose meters, testing miniature endoscopy equipment, calibrating vibrating mirrors, or simultaneously performing functional tests on a batch of two hundred computer hard drives. Below we take a closer look at some of the most important aspects of this design.

#### **26.5.1.1 High-Level Organization**

A major design goal for this system was to serve as a solid, reusable product platform, applicable in a variety of automated test and industrial automation solutions. In general, its implementation is extendable through external and interchangeable software components.

Software components conforming to the interface specified by this framework will, as a consequence, be reusable components as well, which may be stored in a company repository for use in future projects. This component-based approach to software development enables consistent reuse of the components described below.

#### **PATF Engine**

PATF stands for production automation and test framework. The PATF engine component encapsulates the general operations and algorithms required by most applications in this domain, including but not limited to sequencing of operations,

manage requests to and from external devices, manage internal data representation, system configuration, basic interaction with the user, interaction with external software components representing devices under test (DUT), and basic database management using a default format that may be overridden or translated by an external software component (see below). The engine provides two main classes: first, we have the Scheduler class, a *singleton* (Gamma et al. 1995) responsible for assembling the software system at run-time and for orchestrating overall execution. Then, we have the TaskProcessor class, implemented according to the *Command Processor* design pattern (Buschmann et al. 1996). This class provides an arbitrary number of TaskProcessor instances that are responsible for executing the procedures prescribed within AppSequence classes.

### ActiveDevices

These software components are more than just device drivers. They not only implement communication and control over ActiveDevices, but they also include graphical user interfaces and persistent configuration facilities for each ActiveDevice. This set of software components becomes part of a common library of extendable components for use in future projects. These components may be initially built by implementing minimum functionality and then incrementally extend their services, while complying with the ActiveDevice interface as specified by the platform.

Independent ancillary software components describing the current application-specific details include:

1. Application sequence component. Library file that contains code for the application-specific functions or sequence of operations. This component encompasses the main differences that distinguish each product in the family derived from this platform.
2. System configuration Component. Executable file responsible for launching the PATF engine and customizing it for a particular application.
3. Application-Specific GUI Component. Required software component that defines application-specific GUI labels and panels.
4. DutCollection. Optional software component that implements application-specific computation algorithms for devices under test (DUT). Examples are DUT product-specific communications protocols, decision-making algorithms, parameter limits testing, and data-exchange protocol translators.
5. External DB controller. Optional software component that implements interfaces for a specific database, for example, interfacing with remote database servers, generating files that are compatible with other applications like Microsoft Excel<sup>®</sup> or Word<sup>®</sup> processors, and exchanging data with other applications.

Figure 26.9 is a UML component diagram that shows the main software components of the PATF platform and their mapping to the corresponding architectural layers of the framework.

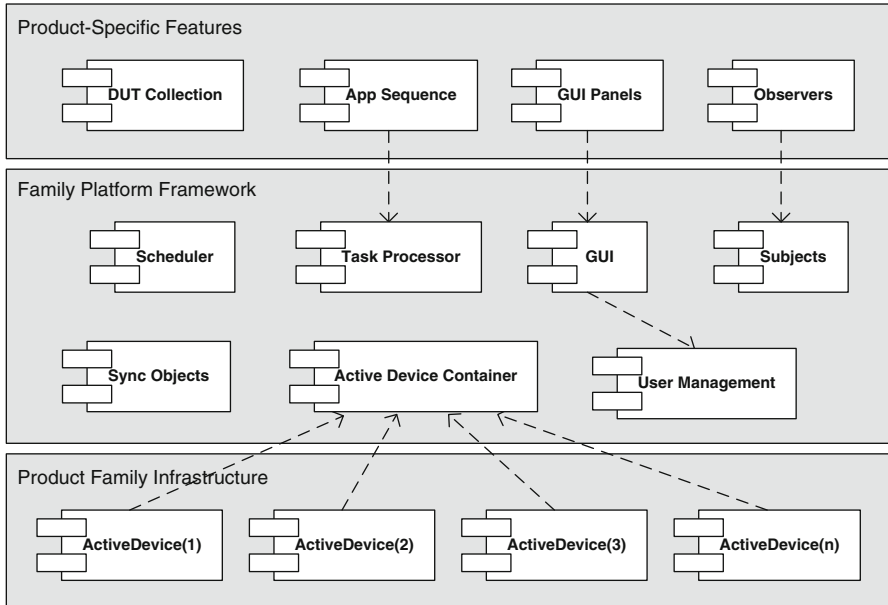


Fig. 26.9 Software components and architecture of the PATF product family platform

### 26.5.1.2 High-Level System Behavior

The proposed domain *abstraction* classifies operation modes of all automated industrial machines as being either manual mode or automatic mode. The system's behavior is driven by a finite state machine, and under these two operation modes, there are six top-level system states. Figure 26.10 shows these states and the events that cause transitions between them.

#### Manual Mode

This mode is used for initializing the machine, modifying operational parameters, and for performing manually controlled operations like jogging a robot, for example. Safety devices are of utmost importance in industrial automation machines, as they play a critical role in preventing operators from getting injured by the machine. In manual mode, however, these safety devices are overridden to enable a qualified technician to perform certain operations that require full control over the machine, as is the case of machine troubleshooting or other maintenance operations. This operation mode is hard coded into the system's engine, but before entering this mode, users are authenticated and only allowed in if they carry the

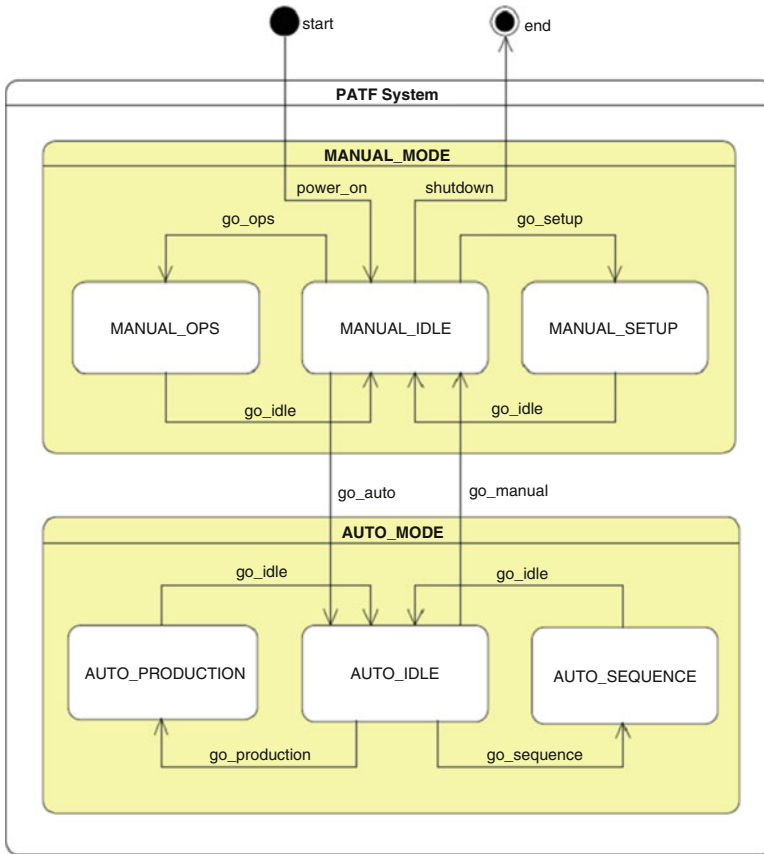


Fig. 26.10 PATF system state diagram

proper credentials, as determined by the User Management component. All safety device signals are ignored and the machine is allowed to run, although displaying a warning message on the user interface through the main GUI component.

### Automatic Mode

This is the normal operation mode of this type of machines. In contrast to manual mode, all system devices and subsystems are engaged, and a built-in safety monitoring system watches for incoming alarms from input/output signals configured as safety devices and automatically calls an emergency shutdown procedure when flagged. Automatic mode also allows users to perform maintenance operations where full speed and safety device enforcement are required, for example, robot coordinate system calibration or product load and unload.

## 26.5.2 *Relevant Design Decisions in PATF*

Let us review the most important design decisions that were put into this framework. A significant feature is that this software application gets assembled at run-time as opposed to compile time. This is possible due to a clear *Separation of Concerns* and *Modularity*. As shown in Fig. 26.9, software components are compiled separately into multiple executables, not as a monolithic application. At the binary level, they are completely decoupled and independent, establishing their relationships and dependencies during execution only. This means that each software component in the Product Family Infrastructure and Family Platform Framework layers is always reused in binary form, not linked as source code libraries at compile time.

Another aspect of significance is that the architecture implements an *Abstraction* of the domain in the shape of behavior, as shown in Fig. 26.10, imposing *Generality* in the behavior of all derived products while still allowing limited customization through the use of externally defined configuration algorithms for each state and substate, which will be explained in more detail later.

The most relevant features, however, revolve around the principle of *Design for Change*, which enables the construction of a wide variety of products. Although they all belong to the same family and share similar characteristics, they can also be applied to a wide range of dissimilar applications across many industries. A more detailed presentation of these features follows below.

### 26.5.2.1 *Design for Change*

For the purpose of illustrating the application of the principle of *Design for Change* (Morales 2013), we now discuss some of the *Evolvability*, *Reusability*, and *Anticipation of Change* features that went into the design of the PATF software product platform.

#### Product Family Infrastructure Layer

Among several needed flexibility features that were observed during the Constraints and Adaptability analysis performed for this project, it was noted that most industrial automated cells would use robots, machine vision, and instrumentation equipment in a consistent manner. The differences between applications were limited to the specific settings and configuration for each infrastructure device, and the sequence of actions that were needed to assemble one product, or test some other. For that reason, all of these components (robots, cameras, instruments, etc.) were packaged as black-box reusable components called *ActiveDevices*. *ActiveDevices* were designed to be implemented in two parts: an *ActiveX* control



and an ActiveX EXE (Microsoft 1994). Both ActiveX components are reused in binary form regardless of the application in which they are used.

Configuration for each ActiveDevice is performed by reading an INI text file containing parameter values for a particular application. When software engineers produce a new application based on the PATF platform, they configure their Infrastructure layer by executing a simple operation of “drag and drop” of ActiveX controls onto the “ActiveDevice Container,” as illustrated in Fig. 26.14, later in this chapter.

As will be seen later on, this approach allowed code reuse of the Product Family Infrastructure layer to reach 100 % across all products in the family.

### Family Platform Framework Layer

It was also observed during the Constraints and Adaptability analysis that a single graphical user interface based on the state machine of Fig. 26.10 could satisfy the needs of all applications in the industrial automation domain, as long as it provided a way to include general-purpose panels in which derived software products could display customized information based on their particular application. This design problem required the provision of a mechanism that could insert customized panels into the Main GUI at run-time.

The solution was to implement the GUI completely as part of the platform framework so it would be reusable in binary form (black box) and add a separate product-specific feature called “GUI panels.” Reusing the GUI along with the system state machine and its related User Management component with its own database represented a major contribution to the overall code reuse ratio, which will be elaborated on later in this chapter.

The family platform framework also supplies generic “Subject” objects which can be instantiated from a GUI panel to work together with an “Observer” object. Observers are described in the next section.

As mentioned previously, the main difference among members of the product family is the sequence of operations they perform on their physical resources to assemble products using a robot and machine vision or the sequence of events when generating electrical stimuli to their devices under test (DUT) on which they perform automated measurements and apply pass/fail criteria. Therefore, a fundamental flexibility need for the platform was the ability to execute one or more simultaneous tasks in a multithreaded environment, effectively implementing an explicit *Separation of Concerns* (Morales 2013). This was achieved by having the platform framework provide generic “Task Processor” objects that would take algorithms encapsulated in a generic object whose class was defined externally, i.e., as a product-specific feature. Thus, a sample application would define a sequence of events for moving a robot about its work cell, operating its gripper, and controlling its speed and joint position, all in a single class of type AppSequence (see below).

Multiple Task Processors can run simultaneously either independently or collaborating through Sync objects (also provided by the platform framework) which enable processors to exchange messages and data, as well as performing a rendezvous that may lead them to alternate paths of execution based on their data exchange, e.g., a test station sending Pass/Fail results to the robot handling the product, which can then drop the product off into a reject bin or pack it into a shipping box, for instance.

### Product-Specific Features Layer

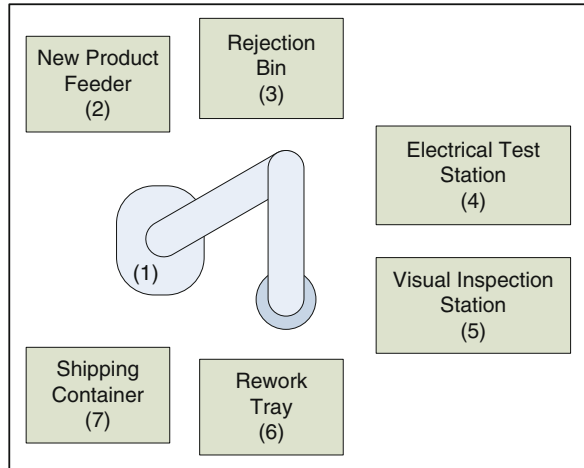
The top layer of the product family architecture houses those features that make individual members of the family truly distinct from each other. The platform framework imposes what is known as *inversion of control* (Morales 2013). This means that all software components in the top layer must comply with the interfaces, protocols, and types expected by the platform framework underneath it. For that reason, this layer consists mostly of code templates. In other words, code reuse in the Product-Specific Features layer is white box. Form is imposed by the software platform, but content is completely determined by the application at hand, i.e., the specific product being implemented.

Whereas Task Processors are generic objects that will execute any task encapsulated in an AppSequence object, the AppSequence class is the product-specific complement to Task Processors. The software family platform specification prescribes some special requirements for the content of these classes, e.g., initialization sequences and exit sequences, but the new application is otherwise completely free to define the sequence of actions on ActiveDevices, to perform any calculations to determine results using external libraries if needed, to exchange data and synchronization points with other sequences (rendezvous), and to update the user interface at any time. During initialization, AppSequences receive references to the ActiveDevice and GUI panel containers in order to have access to the system's resources, and each sequence object announces to the Scheduler with what other AppSequences it needs to collaborate during execution.

GUI panels are plain object containers, which get modified and configured at design-time. All GUI elements like text boxes, list boxes, labels, buttons, tab containers, and frames that are needed by the application can be added freely without any other restriction but the real estate available, which cannot be modified. The number of panels available at run-time is also configured at design-time, being a minimum of one and a maximum of eight. Panels that are not used are hidden at run-time, and users can select which GUI panel is displayed by clicking a button on the screen, independently of the current state of the system (Fig. 26.10). AppSequences can access any GUI widget through the object reference to the container they all have.

Observers are also reused as code templates, and they are typically used to update the contents of GUI panels in an asynchronous manner. Subject and Observer objects are implemented according to the *Observer* design pattern

**Fig. 26.11** A robotic cell based on the PATF product platform



(Gamma et al. 1995), and as described in the referenced book, each Observer subscribes to its corresponding subject during initialization. Subjects are typically updated by AppSequences.

DUT Collection is just another code template that provides a multithreaded execution environment for a collection of application-specific objects defined by a free-form class representing devices under test.

### Final Product Test Example

Further elaborating on the discussion of *Separation of Concerns* as it relates to the implementation of AppSequences, we now present a concrete example to clarify and show how this approach helps to isolate the truly application-dependent features from the common product family functionality.

Figure 26.11 depicts a robotic cell application where we have a robot arm (1) with a gripper to pick up a finished product from the input feeder (2) and take it to an electrical functional test station (4) in the first step of the process. If the electrical test fails, the robot takes the product to a rejection chute (3) that guides the scrapped product to a bin and then picks up a new part from the input feeder. If the product passes the test, it continues in the process and is then taken to a machine vision station (5) for final cosmetic inspection. The visual inspection station consists of a digital camera and its illumination system. If the product fails the visual inspection, the robot takes the product to a tray (6) that is used to ship a batch of rejected products to an external rework station when it's full. If the product passes the visual inspection, then the robot packs the product in the shipping container (7).

This cell is required to keep the count of defective products, count of products sent for rework, and count of good products packed in shipping containers. Additionally, it is required to store the detailed results of all electrical and visual tests,

associate them with the serial number of each product and compute statistical results, and process capability of the robotic cell.

The controlling software for this cell would be implemented as follows:

1. A new class named `RobotSequence` is derived from the `AppProductSequence` class (see Fig. 26.13) to implement the sequence of actions required to control the robot's movements to and from each position in the robot cell, according to the scenario described above.
2. Another class named `ElectricalTestSequence`, also derived from the concrete class `AppProductSequence`, is implemented to execute all the electrical tests at station (4) and to determine the final Pass/Fail result, based on the measurements from each product tested. This class must include a rendezvous point that is exchanged with `RobotSequence` to stop the robot at the electrical test station and pass in the test result, so the robot can proceed to either drop the product in the rejection chute (3) or continue on to the next test station (5). Measurement instruments are all `ActiveDevices`, as described above.
3. The third sequence class, called `VisualInspectionSequence`, is implemented with the image processing algorithms that are needed to perform the cosmetic inspection. These image processing functions are called from an external library acquired from a third party that specializes in machine vision algorithms. The sequence also handles digital inputs and outputs to control the illumination system. This class also implements a rendezvous point to synchronize with `RobotSequence`, stopping the robot in front of the camera and exchanging data representing the inspection results, upon which `RobotSequence` reacts subsequently by placing the product either on the rework tray or in the shipping box.
4. The fourth sequence class is implemented to keep track of the rework tray and automatically swap a full tray with an empty one using a mechanical actuator without having to interfere with the other components of the robot cell. This sequence does not need to synchronize with the robot because they are completely asynchronous and independent. The swapping action is indirectly triggered as the consequence of having the robot place a product in the last available spot on the tray.
5. A `Subject` object is instantiated and assigned an object of type `ProductCount`, which holds the current counts of products tested, rejected, reworked, and shipped.
6. Another `Subject` object is instantiated and assigned an object of type `TestResults`, which holds all the electrical parameter measurements for each product, plus the cumulative statistical results and metrics.
7. Two `Observer` objects are created based on the `Observer` class template. Each of them subscribes to its corresponding `Subject` objects: `ProductCount` or `TestResults`. Each observer is responsible for updating its own GUI panel, which is assigned to them at run-time.

Notice how convenient it is to focus on just one thing at a time (*Separation of Concerns*) when implementing a new application based on the PATF platform, even one as simplified as this example. This advantage becomes more evident as the complexity of the application increases. Additionally, this approach enables teams

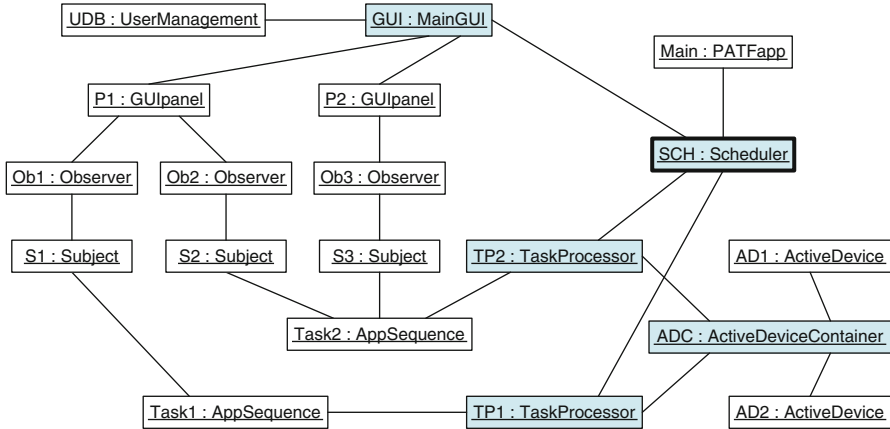


Fig. 26.12 Partial view of the PATF run-time structure (object collaboration)

to divide and conquer the complexity of the problem and to distribute programming work among team members without affecting the system’s consistency, resulting in shorter development times.

**26.5.2.2 Run-Time System Composition**

One of the great benefits of creating a software platform for a family of products is code reuse. Black-box code reuse, in particular, has additional side benefits including robustness, known quality and reliability, and software maturity that improves with each new product that is implemented using the family platform.

Software can be reused in binary form if a multitude of compiled software components are assembled into a customized application at run-time. This is achieved through object composition, as described below. Figure 26.12 is a UML Object Collaboration Diagram showing a partial view of the run-time structure of a generic application built on the PATF product family platform. During initialization, the Main application instantiates the Scheduler, which is the orchestrator of a PATF application, and a chain of events is triggered, at the end of which the whole application is composed and ready to run.

The sequence of events during start-up is roughly as follows:

1. *Main* application starts, indicating a number of parameters and resource locations for the framework’s use. *Scheduler* is instantiated, who takes control over the execution and suspends *Main*. From that point on, *Scheduler* is responsible for instantiating all the necessary objects and assembling them into a complete working application.
2. *Scheduler* instantiates *MainGUI*, which in turn instantiates *GUIpanels* and *UserManagement*. The latter component is responsible for providing user authentication services to *MainGUI* and can use a local database, a remote

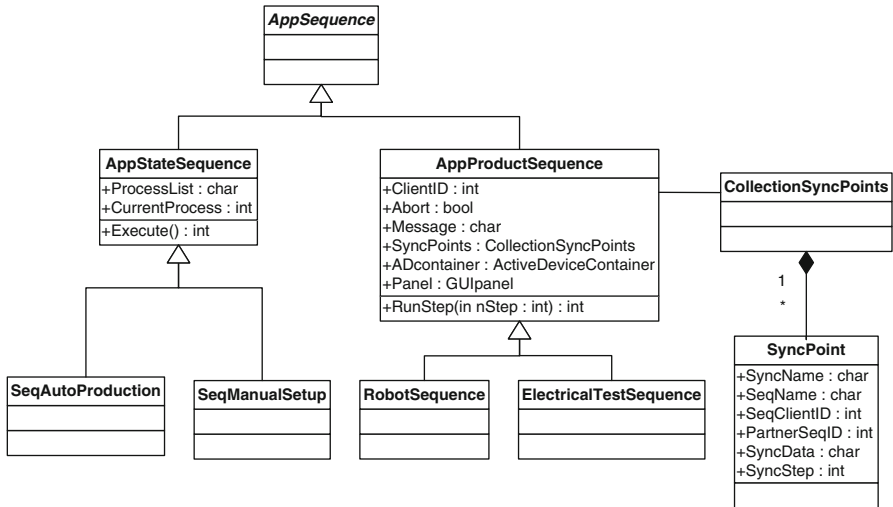


Fig. 26.13 Partial class diagram showing the structure of application sequences in PATF

database connection, or delegate requests to a corporate user repository, details of which are completely hidden from *MainGUI*.

- Once *MainGUI* is up and running, *Scheduler* instantiates the *ActiveDeviceContainer* object, which in turn instantiates all its children components, i.e., all the *ActiveDevices* available to the system in a particular application. At this point, the application objects are all fully assembled in memory, and the system waits in the `MANUAL_IDLE` state (see Fig. 26.10) for user input about the next action.

All applications built on top of the PATF platform behave according to the state machine shown in Fig. 26.10. When the user requests a change of state, the state machine executes the sequence of actions specified in the *AppStateSequence* class corresponding to that state. Figure 26.13 shows a UML class diagram partially showing the structure of *AppSequences*, with a faint reference to the robot cell example described above.

As shown in Fig. 26.13, there are two subclasses derived from the abstract class *AppSequence*: one that defines the actions the system must take during a state transition (*AppStateSequence*) and one that defines the actions to be taken while the system remains in that state (*AppProductSequence*). There are eight *AppStateSequences* that are expected by *Scheduler*, and therefore, they must be implemented by the new product application. These eight classes correspond to each of the low-level system states shown in Fig. 26.10, plus *Initialization* and *Shutdown*. Each *AppStateSequence* class launches one or more *AppProductSequences* as required to be run in their corresponding state. *AppSequences* can be executed in either *Concurrent* or *Sequential* mode, and *AppSequences* may spawn other *AppSequences* in either execution mode. Once children sequences terminate, the parent terminates too. This scheme enables product designers to implement dynamic system behaviors of arbitrary complexity.

When an *AppProductSequence* object starts, it is responsible for attaching themselves to *Subject* objects so they can post the information that is to be displayed on GUI panels after any transformations are made by the corresponding *Observer* objects.

*SyncPoint* objects are instantiated by each *AppProductSequence* object that needs to rendezvous with another *AppProductSequence*. When execution of a sequence reaches the *SyncSteppoint*, the *SyncPoint* object is handed over to *Scheduler*, who takes the request and puts the sequence to sleep until its advertised partner *PartnerSeqID* is ready for rendezvous. At that point, the dormant sequence is awoken, and *Scheduler* swaps the *SyncPoint* objects and sends them back to the partner sequences, thus enabling information exchange between the partners, as described in the robot cell example above.

Run-time object composition as described above maximizes code reuse in binary form, thus ensuring that a more solid product is implemented in a very short time with great technical and economic benefits, as shown in the study of code reuse presented in the next section.

## 26.6 Code Reuse in Product Platforms Vs. Traditional Approach

The PATF product family platform was implemented and initially tested on three different applications within the field of automated industrial manufacturing and testing. More than twenty different products have been implemented and deployed since then.

The first application, System A, is an automated assembly cell for building microelectromechanical system (MEMS) devices. This task requires very high precision, high performance, and high flexibility. It includes a vision-guided SCARA robot, multiple servo-controlled linear actuators, and a range of other industrial automation devices.

The second application, System B, uses the same type of robot in a completely different robot cell platform for automatically testing the same MEMS devices. In this case, the system features multiple automated part feeders that enable uninterrupted operation. This system uses high-precision optical and instrumentation equipment and performs a series of mathematically intensive calculations.

The third application, System C, is a manual version of the robotic tester, using the same optical equipment and mathematical algorithms but manual part handling, interactive operation, and a different set of operator safety policies.

### 26.6.1 Implementation

The system was implemented to run on a Microsoft Windows<sup>®</sup> platform using Microsoft COM technology. Active objects were implemented as multithreaded COM servers (Microsoft 1994), which produce native code for faster execution.

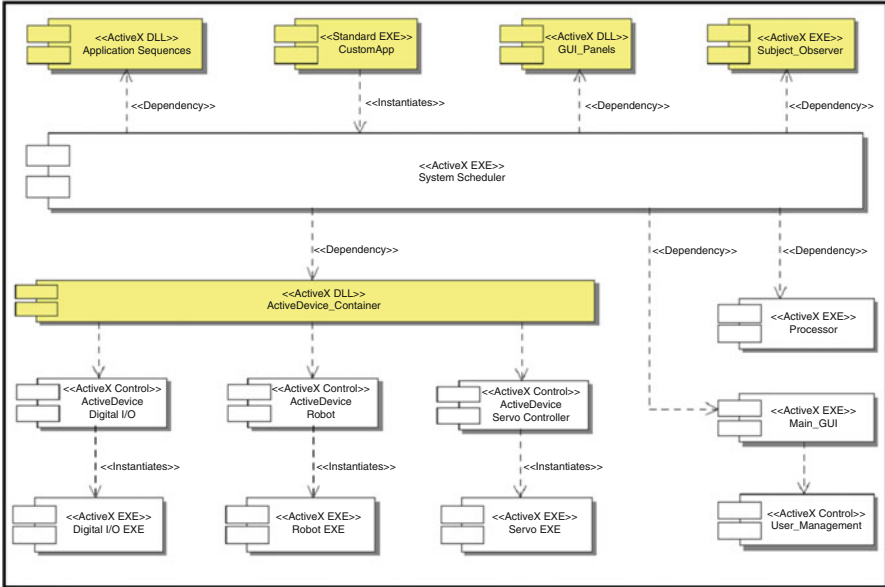


Fig. 26.14 System deployment diagram

In order to simplify the use of ActiveDevices, these components were implemented in two parts: an ActiveX control and an ActiveX EXE executable. The ActiveX control can be dragged and dropped onto an object container, which in turn instantiates an out-of-process COM server in a way that becomes transparent to the application programmer. Figure 26.14 shows a UML deployment diagram with the code components required to implement System A, described above. Similar structures were used to implement Systems B and C.

Software components represented with white symbols are reusable in binary form across multiple applications, and shaded symbols (top row) represent components that must be modified with application-specific code. Therefore, the set of software components that comprise an instance of the system architecture described herein is divided in two subsets according to their relation to the custom application: They are either dependent or independent.

### 26.6.2 Application-Independent Components

Components represented in the UML deployment diagram of Fig. 26.9 are directly mapped from high-level classes in the system architecture. Notice that virtually every software components in the Product Family Infrastructure layer are reusable in executable form. The only exception is ActiveDevice\_Container, which is discussed below.



All of the components in the Family Platform Framework layer are also reusable in binary form across multiple applications, with the sole exception of the Subject\_Observer component, which is also discussed below.

From an economical point of view, components that are reusable in binary form are very valuable assets for an organization, since they are designed, developed, and tested once and can then be used in many future applications using the same devices abstracted by these software components.

### ***26.6.3 Application-Dependent Components***

As naturally expected, most of the components in the top layer (product-specific features) must be customized to fit the target application. Nevertheless, these components are developed based on code templates, where more than 60 % of the source code is reused, with the exception of App\_Sequence, which only reuses a code skeleton or, in other words, just the interface.

Going back to ActiveDevice\_Container, there is virtually no code required in this component, since its only purpose is to package the selected ActiveDevice components into a single compiled component.

As its name hints, Subject\_Observer serves objects of two types: Subject and Observer. There is only one class for Subject, which requires no customization whatever. It is reused at the source code level as is. On the other hand, an Observer class is supplied as an example and code template, which then has to be copied and modified to fit the application at hand. For customization, a typical Observer class requires modifying or writing less than 100 lines of code.

The case for GUI\_Panels is very similar, although most of the modifications are related to user interface objects, like command buttons, data grids, data-bound controls, and labels.

Finally, we get to the executable program CustomApp, which in fact is only a generic name for this component. This program has the sole task of instantiating the System Scheduler with the main settings for the user interface. This is a template project that requires customizing <10 lines of code. In practice, this program is usually compiled with the application name, for instance, RobotCell.exe. Once instantiated, the System Scheduler takes over, and from that point on, CustomApp is actually relegated to the background.

### ***26.6.4 Evaluation***

We now set to evaluate the impact of this approach to software architecture in terms of code reuse and programmer productivity. At the time this project was completed, a survey of the literature on software reuse metrics was made (Chidamber and Kemerer 1994; Price and Demurjian 1997; Price et al. 2001; Washizaki et al. 2003;

**Table 26.1** Summary of product family optimization methods evaluated in study

Projects with similar applications	Total engineering hours
Project A	3,030
Project B	2,290
Project C	2,225
Project D	1,530
Project E (using PATF)	384

Devanbu et al. 1996; Chen et al. 1995; Ferri et al. 1997; Cardino et al. 1997). The objective was to find an appropriate technique that would reflect the benefits derived from reusing enterprise frameworks, where reuse includes both design and executable code. Most proposed metrics addressed class structure, complexity, and static relations that use source code files and other fine-grain elements as inputs. The coarse-grain modularity and functionality of an enterprise framework lay beyond the scope of such metrics. Our conclusion was that new metrics were needed, such that the influence of reusing design, architecture, and other system features located at higher levels of abstraction are also taken into account.

In the case of application frameworks, we think that reuse of architectural design is a major factor in the success of deployed applications using it. When architectural design is reused, the design phase of the software development cycle is greatly simplified, reducing it to a mapping of the application's specific requirements to the different components offered by the standardized architecture, which results in a well-known system organization that streamlines the implementation phase and enhances the quality of the final product. This effect includes important engineering labor savings, which should be quantified in order to take this important benefit into consideration.

### 26.6.5 Methodology

Given this situation, we opted for an informal and simple pragmatic approach to measuring the impact of framework reuse and carry a soft evaluation by using historical data from comparable projects previously completed by the same software development team.

Table 26.1 shows a list of projects previously developed and their respective software development cost in man-hours. All listed applications are very similar robotic cells for automated manufacturing. They all integrate the same robot, the same machine vision system, the same servomotors for linear motion, and other industrial automation tools. Their main difference is that each machine builds a different product. Projects A, B, C, and D were developed using the traditional approach to software development. Project E was implemented using the PATF product family platform. With this information as our starting point, we took the number of engineering hours that were required to design, implement, test, and debug the previous projects and then compared it to the effort required to

**Table 26.2** PATF black-box reuse profile (Project E)

Component	SSI	RSI	Reuse (%)
Scheduler	112,963	112,963	100
Processor	15,961	15,961	100
Main GUI	18,046	18,046	100
User management	14,584	14,584	100
ActiveDevice robot	110,864	110,864	100
ActiveDevice DIO	12,449	12,449	100
ActiveDevice DMM	12,170	12,170	100
ActiveDevice vision	14,762	14,762	100
ActiveDeviceDeviceNet	16,842	16,842	100
ActiveDevice servo	15,308	15,308	100
Total	343,949	343,949	100

Key: *SSI* shipped source instructions, *RSI* reused source instructions

implement a brand new application using the PATF platform. Please note that this table does not include the effort required to design, develop, and implement the platform itself (3,800 h). However, this overhead cost is included in the final evaluation shown in Table 26.5.

## 26.6.6 Results

Let us now characterize the PATF platform and its reusable code base, as fully implemented in the completed product family platform or enterprise framework.

### 26.6.6.1 Code Reusability

Table 26.2 shows the PATF black-box reuse profile, which lists software components that are reused in binary (executable) form. The total number of shipped source instruction lines (SSI) is given as an indicator of the component's size and cost. This is later used as a reference point when quantifying labor savings. Since these components are reused without any changes, the total black-box reuse ratio is 100 %.

Table 26.3 shows the PATF white-box reuse profile. It presents the source code reuse ratio for components that are reused as project and code templates. These templates have to be modified to suit the new application at hand. Naturally, all application-specific code components show low source code reuse ratios. Nevertheless, the overall white-box reuse ratio reaches 20 %, which accounts mostly for abstract classes, type libraries, and other PATF interface elements.

Table 26.4 summarizes the PATF Compound Reuse Profile, which comprises both black-box and white-box reuse ratios, yielding a net shipped code reuse of more than 90 %.

**Table 26.3** PATF white-box reuse profile (Project E)

Component	SSI	RSI	Reuse (%)
Subject	1,597	1,597	100
CustomApp	1,354	620	46
ActiveDevice container	1,288	425	33
GUI_Panels	2,761	863	31
AppSequence	17,620	1,621	21
Observer	2,678	366	14
Total	27,298	5,492	20

Key: *SSI* shipped source instructions, *RSI* reused source instructions

**Table 26.4** PATF compound reuse profile (Project E)

Component	SSI	RSI	Reuse (%)
Black-box reuse	343,949	343,949	100
White-box reuse	27,298	5,492	20
Total	371,247	349,441	94

Key: *SSI* shipped source instructions, *RSI* reused source instructions

**Table 26.5** Comparing PATF vs. traditional approach

Software implementation technique	Total engineering hours
Total project cost using traditional approach (median)	2,258
Total project cost using PATF as the platform (includes 10 % of the development cost of the PATF framework itself)	766
Development cost reduction:	66 %
Productivity improvement:	295 %

### 26.6.6.2 Cost Savings When Using the Platform

It is evident that a reduction in the total implementation effort translates directly into cost savings. Looking at the previous projects, we find that the median development effort for Projects A, B, C, and D is 2,258 man-hours, and the average is 2,269 man-hours.

On the other hand, the total cost for the design, implementation, and test of the PATF framework was 3,815 man-hours. Once ten applications were implemented based on the PATF framework, the investment resulted in a unit cost of approximately 382 man-hours per system. If we add this PATF unit cost to the implementation effort for Project D and take the median as a representative cost value for projects implemented using the traditional approach, we get the results shown in Table 26.5.

The numbers show that projects implemented with the PATF platform cost approximately 1/3 of what it would cost if it were developed from scratch. Likewise, since our cost units are man-hours, the same numbers show that the software team gets the job done three times faster than with the traditional approach.

Cost savings in engineering effort are an important piece of information to economically justify the greater investment needed to develop reusable software architectures and code within an organization.

Keep in mind, however, that software product platforms, or enterprise application frameworks, are hard to design and the challenge should not be taken lightly. It requires an expert software architect with deep knowledge of the domain, supported by a team of software professionals, and the full commitment of the organization to make the significant investment that requires such project. Nevertheless, the payoff of a successful software product platform is significant and rewarding in the long term.

### 26.6.6.3 Other Observations

Since PATF implies that a complete set of design decisions have been made for software developers of future projects that use it, they did not have to spend any time at all designing their new applications.

Likewise, developers had to spend no time thinking of how to identify, create, or abstract new modules or interfaces, since everything is fixed a priori in the framework. Functional policies that are typical to most applications within the particular domain are also implemented in the underlying design and platform. Therefore, the only task left for them to do was to focus on the particular functional details that made the target application unique, i.e., the specific algorithms of their new assignment.

Due to the phenomenon of inversion of control exhibited by an application framework, programmers are forced to write application-specific code strictly following the guidelines and interfaces prescribed in the framework design. The resulting code was more homogeneous and standardized across the team, as compared to code produced for previous projects. This effect is an additional benefit, since it also helps to improve the software product maintainability.

It is important to mention, however, that at the beginning, it was a difficult task for software developers to grasp the system model and the new inverted-control programming style where the family platform framework was in charge, not them. Inflexibility of the interface design and the multitude of stand-alone external components required by the system as mandatory also demanded additional adjustments to the typical programmer mindset. Nevertheless, after the learning period was over, programmers acknowledged the benefits of the new structured development based on the new product platform.

### 26.6.6.4 Code Reuse Conclusion

In agreement with previous reports (Fayad et al. 2000; Fayad and Schmidt 1997; Schmidt and Fayad 1997; Yassin and Fayad 1999; Poulin et al. 1993), the effect of using a software product platform for the industrial automation and test domain was

positive, having achieved an estimated level of executable code reusability above 90 % and cost savings of about 60 %. Productivity of software development teams was improved, although the learning curve could be steep for some programmers.

Reusing both design and code yields multiple benefits in terms of cost, new product development lead time, and robust quality of the production code due to extensive testing at each new product iteration. Homogeneity and consistency among different members of the product family makes maintenance and evolution feasible, efficient, and cost-effective.

## References

- Alur D, Crupi J, Malks D (2003) *Core J2EE patterns*, 2nd edn. Prentice Hall, Upper Saddle River, NJ
- Booch G, Rumbaugh J, Jacobson I (1998) *The unified modeling language user guide*. Addison-Wesley, Reading, MA
- Brooks FP (1975) *The mythical man month*. Addison-Wesley, Reading, MA
- Buschmann F et al (1996) *Pattern-oriented software architecture: a system of patterns*. Wiley, Chichester
- CACM (1997) Special issue on object-oriented application frameworks. *Commun ACM* 40 (10):32–87
- Cardino G, Baruchelli F, Valerio A (1997) The evaluation of framework reusability. *ACM SIGAPP Appl Comput Rev* 5(2):21–27
- Chen YF, Krishnamurti B, Vo KP (1995) An objective reuse metric: model and methodology. In: Schäfer W, Botella P (eds) *ESEC '95: 5th European software engineering conference*, Sitges, Spain, Sept 1995
- Chidamber SR, Kemerer CF (1994) A metrics suite for object oriented design. *IEEE Trans Soft Eng* 20(6):476–493
- Clemens P, Bachmann F, Bass L, Garlan D, Ivers J, Little R, Merson P, Nord R, Stafford J (2011) *Documenting software architectures*, 2nd edn. Addison-Wesley, Reading, MA
- Cockburn A (2000) *Writing effective use cases*. Addison-Wesley Professional, Reading, MA
- Daigneau R (2011) *Service design patterns: fundamental design solutions for SOAP/WSDL and RESTful web services*. Addison-Wesley Professional, Reading, MA
- Devanbu P, Kartsu S, Melo W, Thomas W (1996) Analytical and empirical evaluation of software reuse metrics. In: Dieter Rombach H et al (eds) *ICSE '96: 18th international conference on software engineering*, Berlin, Germany, 25–29 Mar 1996. *Proceedings IEEE Computer Society* 1996, pp 189–199
- Douglass BP (2002) *Real-time design patterns: robust scalable architecture for real-time systems*. Addison-Wesley Professional, Reading, MA
- Douglass BP (2011) *Design patterns for embedded systems in C*. Newnes, Boston, MA
- Fayad ME, Schmidt D (1997) Object-oriented application frameworks. *Commun ACM* 40 (10):32–38
- Fayad ME, Schmidt D, Johnson R (1999) *Building application frameworks: object-oriented foundations of framework design*. Wiley, New York, NY
- Fayad ME and Johnson R, (1999) *Domain-Specific Application Frameworks: framework Experience by Industry*. Wiley, New York
- Fayad ME, Hamu DS, Brugali D (2000) Enterprise frameworks: characteristics, criteria and challenges. *Commun ACM* 43(10):39–46
- Ferri RN, Pratiwadi RN, Rivera LM, Shakir M, Snyder JJ, Thomas DW, Chen YF, Fowler GS, Krishnamurti B, Vo KP (1997) Software reuse metrics for an industrial project. In: Bieman

- et al (eds) METRICS '97: Proceedings of the 4th international software metrics symposium, Albuquerque, NM, Nov 1997, pp 165–173
- Fowler M (2002) Patterns of enterprise application architecture. Addison-Wesley Professional, Reading, MA
- Fowler M (2003) UML distilled, 3rd edn. Addison-Wesley, Reading, MA
- Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: elements of reusable object-oriented software. Addison-Wesley, Reading, MA
- IEEE (2011) ISO/IEC/IEEE Std 42010-2011 systems and software engineering: architecture description, international standard
- ISO (2011) The ISO architecture group maintains a survey of all known architecture frameworks as reference material in their ISO/IEC/IEEE Std 42010. <http://www.iso-architecture.org/ieec-1471/afs/frameworks-table.html>. Accessed Jul 2012
- Jacobson I, Christerson M, Jonsson P, Overgaard G (1992) Object-oriented software engineering: a use case driven approach. Addison-Wesley, Reading, MA
- Johnson RE (1992) Documenting frameworks using patterns. In: Pugh J (ed) OOPSLA '92: ACM conference on object oriented programming systems, languages and applications, conference proceedings, Vancouver, BC, Canada, Oct 1992
- Johnson RE (1997) Frameworks=(components+patterns). Commun ACM 40(10):39–42
- Microsoft (1994) COM and COM+Technology reference papers. <http://msdn.microsoft.com/>
- Morales CO (2013) Chapter 21: Design principles for reusable software platforms. In: Simpson T, Jiao R, Siddique Z, Holtta-Otto K (eds) Advances in Product Family and Product Platform Design: Methods and Applications, Springer, New York, NY
- Object Management Group (2011) The unified modeling language UML, ver. 2.4.1, Object Management Group
- Poulin J, Caruso J, Hancock D (1993) The business case for software reuse. IBM Syst J 32 (4):567–594
- Price MW, Demurjian SA (1997) Analyzing and measuring reusability in object-oriented designs. In: Berman AM (ed) OOPSLA '97: ACM conference on object oriented programming systems languages and applications, conference proceedings, Atlanta, GA, Oct 1997, pp 22–33
- Price MW, Needham DM, Demurjian SA (2001) Producing reusable object-oriented components: a domain-and-organization-specific perspective. In: Proceedings of ACM symposium on software reusability SSR '01, Toronto, ON, Canada, May 2001, pp 41–50
- Rumbaugh J, Jacobson I, Booch G (1998) The unified modeling language reference manual. Addison-Wesley, Reading, MA
- Russell S, Norvig P (2009) Artificial Intelligence: a modern approach, 3rd edn. Prentice Hall, Upper Saddle River, NJ
- Schmidt D, Fayad ME (1997) Lessons learned building reusable OO frameworks for distributed software. Commun ACM 40(10):85–87
- Schmidt D, Stal M, Rohnert H, Buschmann F (2000) Pattern-oriented software architecture, vol 2: patterns for concurrent and networked objects. Wiley, Chichester
- Washizaki H, Yamamoto H, Fukuzawa Y (2003) A metrics suite for measuring reusability of software components. In: Proceedings of the 9th IEEE international software metrics symposium (METRICS 2003), 3–5 Sept 2003, Sydney, Australia, pp 211–223
- Yassin AF, Fayad ME (1999) Chapter 29: A survey of object-oriented application frameworks. In: Fayad ME, Johnson R (eds) Domain-specific application frameworks. Wiley, New York, NY, pp 615–632
- Zachman JA (1987) A framework for information systems architecture. IBM Syst J 26(3):276–292

# Chapter 27

## Customer Needs Based Product Family Sizing Design: The Viper Case Study

Cassandra Sotos, Gül E. Okudan Kremer, and Gülşen Akman

**Abstract** This study explores the issue of optimizing the size of a product family as well as designing the product variants of the family in the context of designing a stand for a unique electric violin known as the “Viper.” The study uses collected customer data in order to identify the optimal number of design variants in the product family, generate design alternatives, assign each design to the appropriate customer type, and verify the outcome. The methodology begins with data collection through a survey of Viper players which is analyzed using K-means clustering analysis and segmentation in order to determine the optimal number of variants in the product family. This analysis also defines each customer group and each group’s specific customer requirements. Quality function deployment (QFD) is used to fit design concepts (generated by the requirements and specifications) to synthesized design variants and assign to a customer group. Then, analytic network process (ANP) is used to substantiate the outcome of the study by further verifying each Product Family Member-Customer Group mapping as well as the number of variants. This is achieved by simultaneously fitting the generated design concepts to customer requirements and customer groups through ANP. The ultimate goal of this work is to provide a simple, easy to understand, easy to reproduce, and customizable method for making product family size and design decisions in any industry.

---

C. Sotos • G.E. Okudan Kremer (✉)  
The Pennsylvania State University, University Park, PA, USA  
e-mail: [gek3@psu.edu](mailto:gek3@psu.edu)

G. Akman  
Kocaeli University, Kocaeli, Turkey



## 27.1 Product Family Sizing Design

Product family and product platform design problems are topics commonly written about in the academic and business worlds and include issues and viewpoints presented from many fields. Design decisions can be based on a number of factors and are often simultaneously explored by many areas of study and expertise: business and finance, supply chain and operations research, engineering, marketing and computer science, etc. In many cases, these areas have very different, and sometimes conflicting objectives during each stage of the design phase. Each objective comes with its own set of trade-offs that need to be considered. Identifying the most important factors and achieving the correct balance between the differences in objectives for each individual problem can become very complex. The methodologies presented, in general, address issues of commonality, flexibility, and cost.

More than ever, meeting the rapidly changing and demanding requirements of customers is of the highest importance to product designers and executives around the globe. In recent years, customers have grown accustomed to their own purchasing power granting them nearly-instantaneous product gratification: perfectly matched and customized to their individual requirements and specifications with high quality and, most importantly, right when they need it. This trend is clearly displayed in the industry of application development for mobile devices and computers. Customer perceptions and expectations from companies are moving in the direction of the previously mentioned trend across industries in general, reflected in the continuous enhancements and improvements upon the current technology for manufacturing facilities. To survive in such an intensely competitive and volatile market, firms, small and large, must adapt their practices to be able to meet the requirements of a breadth of customers with differing requirements while maintaining low retail prices and without damaging profits.

Research shows that when a customer is presented with product options that meet their individual desires and requirements, aesthetically and/or functionally, they are likely to pay more for these products than for lesser varieties (Tseng and Jiao 1996). One method of meeting differing requirements within a company's customer base is to offer a variety of products such as in a product family (Salhieh 2007). In today's market, it is becoming harder and harder to meet customer requirements with a single product as many customers want a choice of several options as well as rapidly released new updates. An alternative to cope with these demands is mass customization, the production of individually customized and highly varied products (Meyer and Lehnerd 1997). The issues of demanding and varying customer requirements and how to address these issues are a common theme in recent publications as they relate to both mass customization and product family design (e.g., Jiao et al. 1998; Jiao and Zhang 2005; Li et al. 2008), where the authors stress the importance of incorporating more directly customer-related input into the design process in research as well as in industry. However, the aforementioned issues are not typically seen as the focus of most publications.

One way to meet the demands of a larger group of customers is to offer a variety of products such as in a product family. However, the issue of potential product cannibalization can be a problem. In this context, cannibalization can be described as one product variant diminishing the returns (e.g., market share and profits) of another. For example, introducing a DVD player with new technologies into a product family can increase its own revenue, but it may lower the sales of other DVD players already in production by the same manufacturer. Due to this concern, when a company is planning to launch a product variant to market where other variants exist, customer fit and profitability should be considered across the product family and at the individual variant level. Conversely, if the available variants of the product family marketed by a company do not respond to the actual customer needs, competing companies can infiltrate the market by launching new variants. In other words, neither having more than needed nor less than needed product variants is beneficial for a company. Many authors acknowledge that the number of variants in the product family is an important factor to consider, but in most cases it is an afterthought, used as a metric to describe a family once it is designed or a by-product when trying to decide upon another factor. Those who do directly approach choosing the optimal number of variants ( $N^*$ ) do it in a very complex manner that requires a large amount of resources, historical customer and engineering cost information, and breadth of engineering knowledge. Many of the methods currently in literature are notably complex and would not be on a level that most small to mid-sized companies could accommodate from within their organization's resources. The methods that we propose here seek to simplify the process of starting the design of a product family by providing an easily reproduced (and easily customized to the particular industry or business) method of deriving the optimal number of variants in the product family ( $N^*$ ) and then mapping customer requirements to those products utilizing one all-encompassing set of data. While this method requires notable effort in customer market intelligence, the information collected is used throughout the methodology. The methodology should, using customer requirements, provide a solid and simple starting point for new product family development that effectively builds a foundation for the rest of the design processes involved.

Issues explored in product family design are similar to those of product platform design, including cost, flexibility, and market demands. However, the topics considered also span into product variety, performance, customization, and manufacturability. Some researchers discuss the different strategies and processes involved in designing a product family that is based on a modular platform. One of the most called for topics in industry and academic research is incorporating more customer-direct data into the design process. An example of work that heavily utilizes customer data and is similar to what we are proposing is described in a method by Tucker and Kim (2009). The methodology is based on customer data collection and analysis techniques, in particular decision tree classification. The authors collect customer data, segment it, and then use that information in the concept generation process so that they may define what functionalities customers require. The thought behind this method of concept generation is to optimize the

resources and time spent on concept generation and selection when one would normally be evaluating much larger lists of design alternatives. Once designs are selected for the products, a concept cost/profit multilevel optimization model is constructed and solved. Out of this model comes a number of variants, denoted “N,” but it is based on the engineering costs and profits rather than the original customer data. Although choosing the number of variants for the family is a part of this methodology, its main focus is to design the variants based on the collected customer data.

Along the same lines, Zha and Sriram (2006) present a method which is knowledge-based and emphasizes the importance of incorporating several areas of knowledge into all steps of the design process as well as modeling that knowledge along the way. Considerations of product variety are one of the factors mentioned as a way to apply knowledge-based methodologies. In terms of applying this to product family sizing, the authors briefly mention two metrics that are related to the number of product variants: market efficiency  $\eta_M = N_{tm} / NM$  (number of targetable market niches)/NM (total market), and investment efficiency  $\eta_I = C_m / N_v$  (manufacturing equipment costs)/Nv (number of product varieties). These metrics provide a way of evaluating a product family after the fact. They do not explicitly tell designers when or how to create a new family. Also along the lines of incorporating customer data, Kumar et al. (2009) propose a methodology called “Market-driven product family design” that incorporates market data (similar to our case) with manufacturing and engineering costs. The methodology begins with market segmentation and demand estimates and then furthers to cost and performance models. One step of the methodology is product family optimization, and it addresses finding the optimal number of product variants. The number of product variants is found simultaneously with the number of platforms in the family (commonality decisions). How many platforms to use is the greater focus of the study. This methodology is not exclusive for new product design; it includes preexisting markets in the segmentation. The authors propose multiple platforms for the product variants rather than building upon one platform. Although this is a very elegant model that covers many decisions simultaneously, it is complex and requires a lot of information. In situations where there are limited resources and information about the market or where it is not anticipated that the company wishes to work through multiple platforms, this methodology may not be appropriate.

Some researchers have focused on how to expand family products through modularization. They have used function structure and the physical principle of components or a product itself to set modularity rules, by which they have generated possible new modules and potential product matrices (Dahmus et al. 2000). Others have used modular architecture and module commonalization combined with quality function deployment (QFD) to build function and module structures for target products (Fujita et al. 1999). By mapping customer needs to functions and manufacturing in receiver circuits, researchers believe they can extend product variety and reduce costs (Fujita et al. 1999). Through the implementation of such an approach, scholars have made minor adjustments such as

performing successive quadratic programming (SQP) to derive new modularity elements (Fujita 2002). Similarly, a way to optimize a product family is to define what (if any) is the most efficient use of a product platform or at least what components should or should not be common throughout the family. Huang et al. (2008) address product family optimization by defining which variables should be common throughout the product family and which of the variants should include the said variables. They accomplish this by using a multi-objective genetic algorithm and introduce a commonality index which is utilized in the model. In this work, the product variants are optimized individually but simultaneously. The simultaneous optimization is described as a way to preserve individual variant performance while still considering commonality and avoiding great differences in the variants themselves. In this methodology, it is assumed that the number of product variants ( $np$ ) is already known when entering into the problem and it does not provide a way of choosing  $np$ . The authors use the multi-objective genetic algorithm as a way to control and define commonality across the product family with  $np$  product variants and  $nv$  design variables.

There are many instances of research where the number of variants in the product family is a part of the process that confirms the “N” (number of variants) after the fact or is a predefined input into the process. One example is in the work by Bhandare and Allada (2009), an approach to product family design that begins with known variants and then based on those variants, selects the optimal number of platforms and which platforms to use. They suggest multiple scalable platforms in the family to reduce the degradation of individual variant performance which they say is common when building upon one platform. To achieve optimal platform selection, the authors propose an optimization problem that minimizes costs to the manufacturer while considering the demand constraints of each individual variant. This model also considers the cost of performance loss of each variant due to platforming in addition to the manufacturing costs. This methodology demands intensive information gathering for the model of each variant: demand data, product specifications, technical specifications and requirements, defining variables that affect performance, etc. The optimization problem allows for the identification of subsets of scalable and common variables. Beyond this, it also calls for the simulation of platform cases—using 1 through  $m$  platforms ( $m$  platforms would mean that there is a platform for each variant) and evaluating the implications of each case. Given the results of the optimization combined with the evaluation of each of the cases, the authors are able to select the optimal number of platforms to derive the variants from. The authors demonstrate their proposed methodology by applying it to a product family of axial piston pumps. In this case, the demand information is known a priori and there is a significant amount of technical information (i.e., requirements and specifications) to be included in the optimization problem. This methodology is applicable to many technically complex problems; however, it may be difficult to scale up the optimization section as it has been currently developed for use with less complicated product families.

## 27.2 Background Information and Problem Description

### 27.2.1 *Wood Violins and the Viper Electric Violin*

The Viper electric violin is produced in the USA by a privately owned company, Wood Violins, and has been available to the public for around 15 years. The Viper offers violinists, violists, and other instrumentalists seeking an electric instrument a plethora of options which are typically not available in configurations from other manufacturers. Some of these options include custom finishes, the addition of up to three strings (extending the total number of strings to seven, three strings lower than those of a traditional four string acoustic violin), and the possibility of adding frets similar to an electric guitar. The benefits and trade-offs of adding or excluding these options are determined by the customer's personal preferences, and almost all Vipers are currently customized and made to order. The Viper's suggested retail price ranges from \$2,500 to \$5,500 depending on the features included and excluded in the ordering period. A side view of the Viper can be seen in Fig. 27.1. The type of customers who currently purchase and own Vipers ranges from young students to full-time professionals. Wood Violins offers a whole line of instruments from student to professional models with the Viper representing the most "luxury," high-end product in the line. It is also considered one of the flagship products of the company. The shape of the Viper resembles a "flying-V" electric guitar and is very different from a traditional acoustic violin. The shape provides aesthetics as well as functionality by lending itself to the Wood Violins-patented chest support system, which makes the instrument completely self-supporting. The need for the



**Fig. 27.1** Side view of Viper shown with strap and extended chest support (<http://www.woodviolins.com>)

chin rest and shoulder pad is eliminated and the user's head, neck, and arms are free and overall mobility and flexibility is increased. This is achieved by utilizing a paddle and a strap that stabilizes the instrument on the musician's body.

Although it most definitely adds value to the instrument, the Viper's shape precludes it from being paired with many commonly used accessories for acoustic violins and violas such as instrument stands and instrument cases. Wood Violins currently carries a custom shaped case for the Viper electric violin, but there is no available instrument stand that safely suits the shape of the Viper.

Instrument stands are a commonly used accessory for all musical instruments ranging from stringed instruments, brass, and wind instruments to electric instruments like guitars and bass guitars. A stand maintains the safety and security of the instrument by holding it upright and stable while still keeping it very easily accessible in performance or home practice situations. Stands are also very commonly used by multi-instrumentalist performers who have limited stage space but need to switch between several instruments quickly and very frequently. There are no stands on the market that are specifically made for Viper electric violins. As of now, Viper players who actively seek and use an instrument stand are using the next best alternatives to a custom-fitted Viper stand. These are guitar stands, which do not meet all the geometric and safety needs of the Viper.

Since almost all Viper players are multi-instrumentalists (almost all play at least the acoustic violin or viola and many others play mandolin or guitar), and there is an existing common use for instrument stands for other musicians, there is also most likely need for a Viper stand. Combining this knowledge with the fact that the Viper is a relatively expensive product that has an existing and growing customer base, it is also likely that customers would be willing to seek out and purchase a new accessory for their instrument, and this possibility would be worth investigating. Even though the type of production and materials may change in new generations of the Viper, the geometry of the instrument would not be changed, so the designs resulting from this study would remain relevant.

### **27.2.2 Objectives**

To address the task of designing a new family of stands for the Viper, we propose a methodology to choose the number of variants in a product family, assign components to each family "member" to define a product setup for each variant, and verify the results.

This methodology is demonstrated by the design of a product family of stands for the Viper electric violin. It is hypothesized that there are several distinct customer groups of Viper players that have equally distinct needs when purchasing accessories such as a stand for their instrument. It has also been identified that there is a customer need for a Viper instrument stand. The goal of the methodology is to determine the optimal number of variants in a product family ( $N^*$ ) of Viper stands

corresponding to the number of distinct groups of customer types indicating their needs and preferences as well as generate distinct designs for each group.

A concurrent goal of the methodology is to provide a guideline for sizing and designing a product family that is simple, easily reproducible, and easily customizable across a range of industries and problem types. The methodology provides a solid foundational starting point for the design of a product family that is easy to understand and can be applied by designers with business and engineering backgrounds alike.

### 27.3 Methodology

The methodology begins with the creation of a survey (or any medium of customer data collection—this phase is customizable to the problem type and the resources available), validation and testing of the survey, and deployment. It should be noted that the first step in this methodology assumes that the objectives of the study have been identified a priori so that meaningful questions can be written for the survey. Once the data has been collected, the responses are cleaned and categorized when possible and, if not already in numeric formats, transformed to have numeric representation. With this data set, the K-means clustering method is implemented along with a validity measurement to identify the number of groups in the product family ( $N^*$ ). Once the data is segmented, the original customers are mapped to each segment, or customer group, so that new descriptive statistics can be calculated for each customer group. From this segmented information, design requirements and specifications are generated for each individual group. Concurrently, design alternatives for the components of the variants in the product family are generated based on what currently exists on the market as well as newly generated ideas. These components are ranked with respect to how well they meet certain design requirements and matched with the customer group components using quality function deployment (QFD). This is used to generate actual design variants by combining the components selected for each group. Finally, the findings are verified by implementing analytic network process (ANP). The flow of the methodology is shown in Fig. 27.2.

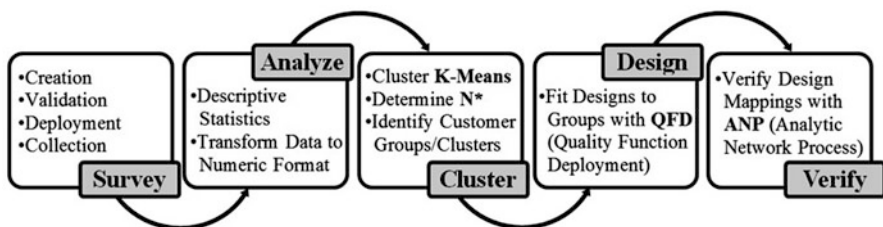


Fig. 27.2 Methodology flow

### ***27.3.1 Survey Creation and Deployment***

A survey was created as a medium to collect data from current users of the Viper electric violin. The survey was designed in such a way that the person taking the survey would not know that it was related to the creation of a stand. Instead, the survey sought to define the type and frequency of interactions with the Viper in different settings where the user might want to use a stand: home, performance, and travel, and in doing so answer the designers' questions about potential product needs. Creating the survey in this manner prevents biased answers from the users. Instead of asking the user a direct question about a stand such as What are the most important qualities you need in an instrument stand while practicing?, we ask a series of questions about the way they practice: what, when, how, etc., and we are able to infer an answer to the first question based on those answers. The survey was divided into six sections: (1) About You, (2) Specifics about Viper Playing, (3) Playing your Viper at Home, (4) Performing with your Viper, (5) Traveling with your Viper, and (6) Supplies and Accessories for your Viper.

The survey allows us to profile Viper players, sort them into groups of similar users who express similar needs and preferences, and identify what percentage of the total market the group holds. For instance, there may be a notable group of professional, full-time performers who also play at least one other instrument, travel frequently, and play at home very infrequently. Members of this group may be most concerned with flexibility and portability. There may also be a group of amateur/recreational Viper players who play only one other instrument, perform very irregularly and casually if at all, and never travel. Members of this group may be less concerned with portability and more concerned with cost. The individual sections of the survey ask questions that let us know specifics about the types of interactions members of these group experience which in turn allow us to identify their preferences and needs.

The survey was tested by a group of pilot users ( $n = 4$ ) and then deployed via the Internet and through Wood Violins' electronic mailing list. During the 3-week surveying period, 111 valid responses were collected. Participants were mainly from the USA, but some international Viper players also responded.

### ***27.3.2 Data Cleaning and Interpretation***

A very important part of this process was to select the correct questions to be clustered. The selected question and the corresponding answer choices are those that indirectly or directly influence customer needs and preferences when dealing with an instrument stand. The original survey contained many more questions, but some of the questions were inserted for the purpose of unrelated research to be used for later projects such as marketing, or the importance of the answers were already captured by another question that was included for the clustering analysis.



Many of the responses needed to be transformed to numeric values for the clustering analysis. The questions were classified into four categories: numerical questions, nominal single answer questions, nominal multiple answer questions, and ordinal questions. For example, the answer choices for the question, “What is your highest level of formal music training?,” were originally in the format of none-self taught, pre-college private lessons, adult private lessons, undergraduate music minor, undergraduate music major, graduate music degree, and conservatory trained. The answer choices were identified as ordinal and were relabeled in the format 0–6: 0 (none-self taught), 1 (pre-college private lessons), 2 (adult private lessons), 3 (undergraduate music minor), 4 (undergraduate music major), 5 (graduate music degree), and 6 (conservatory trained). This process ultimately led to transforming 25 original survey questions into 61 variables for the analysis.

### **27.3.3 Cluster Analysis**

After converting all of the data to numerical values, we implemented the K-means clustering method and used a validity measure to select the optimum number of clusters. K-means clustering (MacQueen 1967) is a widely used method to partition a data set into  $k$  groups. It starts with the selection of the  $k$  initial cluster centers and then iteratively refining them as (1) each instance  $d_i$  is assigned to its closest cluster. (2) Each cluster center  $C_j$  is updated to the mean of its instances. The algorithm converges when there is no further change in assignment of data instances to the clusters (Wagstaff et al. 2001).

This method was chosen after study of the current literature because it is simple to understand and implement for all types of professionals in varying fields of engineering and business. It was also chosen because it provides an up-front foundational starting point for the design of the family in regard to the number of variants, where most current methodologies examine  $N$  variants as an afterthought in the form of a metric to describe the current state of the family. It also allows for greater flexibility for different business and engineering objectives of management. The K-means clustering method requires that the  $K$  (number of clusters) is preselected. Once  $K$  is selected, the data is clustered into  $K$  groups by using a Euclidian distance formula. Since the  $K$  requires preselection, completing K-means clustering analysis alone will not provide an optimum number of clusters. Each scenario (2 clusters, 3, 4, 5, and so on) has to be calculated independently. The scenarios for  $K = [2, 3, 4, 5, 6, 7, 8, 9, 10]$  were all computed for this analysis.

### **27.3.4 Validity and Selecting the Optimum Number of Clusters**

As was previously mentioned, the information obtained from only calculating scenarios  $K = [2, 10]$  does not provide specific insight about the optimum number of clusters. Authors Ray and Turi (1999) present a method of selecting the optimum

number of clusters when using K-means by using a measure they call validity. The validity measure provides a way to compare the said scenarios and aids in making a decision about the optimum number of clusters. Validity is defined as the ratio between the intra-cluster distance measure (the distance between points inside one cluster) and the inter-cluster distance measure (the distance between clusters). When validity is small, it implies that the points within one cluster are very close to one another (i.e., small intra-cluster distance) and also very far away from the data points in other clusters (i.e., large inter-cluster distance).

The authors also suggest that instead of selecting the purely minimum validity value, one should identify the local maximums and explore the points around each maximum to find the most improvements or bend when moving between K values.

### ***27.3.5 Discovering the Customer Groups***

Once the optimum number of clusters has been identified, the group membership can be extracted from the original analysis. With new customer groups divided from one another instead of the original overall customer group, new descriptive statistics can be calculated for each group, and the customer types can begin to be discovered along with their needs and preferences. The group membership is linked to the original respondent identification number, and therefore all of that respondent's information from the original survey (including the questions that were omitted from the clustering analysis) can be accessed and analyzed for further research.

### ***27.3.6 Matching Designs to Customers with QFD***

In the final steps, quality function deployment (QFD) is used to fit generated design concepts to each clustered customer group. Since it was proposed in 1966 by Dr. Yoji Akao and Shigeru Mizuno, QFD has been applied in numerous industries. It connects the voice of customers (VOC) to the product designers. It is a systematic way to transform the customers' needs for a product into prioritized technical measures that can be further deployed to develop process and production plans. With QFD, user expectations (the "Whats") are related to design and production-related parameters (the "Hows"). This process is represented by a succession of double entry "Whats/Hows" tables, allowing the correlations between entries to be identified and prioritized (Chiu et al. 2008).

The benefits of QFD are well known. Chan and Wu (2002) classified the benefits into qualitative and quantitative benefits. Three major qualitative benefits are enhanced customer orientation, effective product development, and improved communications and teamwork. For the quantitative benefits, Bicknell and Bicknell (1995) outlined

the estimated tangible benefits that are common when QFD is properly used as 30–50 % reduction in engineering changes, 30–50 % shorter design cycles, 20–60 % lower start-up costs, and 20–50 % fewer warranty claims. Given its benefits, QFD is chosen for application here.

For the QFD application, a concise list of the most important requirements for each group is generated based on the customer responses, habits, and needs demonstrated by the original survey questions. These needs are linked to a number of possible concepts to be included in the final designs. The final designs are selected by how well each individual concept is able to meet a group's specific needs.

### ***27.3.7 Verifying Consistency of the Results with ANP***

As per the steps of the QFD application, it is not possible to verify the consistency of the fit judgments. Accordingly, analytic network process (ANP) is used to simultaneously fit generated design concepts to customer requirements and customer groups. ANP is the more general form of the analytical hierarchy process (AHP) used in multi-criteria decision making (MCDM) to ease limitations related to dependency across variables (Huang et al. 2005). While AHP ensures a hierarchical decision framework by employing a unidirectional relationship within the range of decision-making levels, ANP provides modeling more complex interrelationships among the decision levels and sub-criteria (Meade and Sarkis 1998). ANP is also recognized as the systems-with-feedback approach. A supermatrix will be created by linking interdependencies and feedback loops in the model. The supermatrix arranges the relative weights in individual matrices, and then a new overall matrix is constructed with the eigenvectors of the arranged relative weights (Meade and Sarkis 1998). The supermatrix has limiting impact which provides the priorities of the factors in the network and it is computed for each network (Saaty 2001). In ANP, weights are calculated by means of three matrices: (1) the unweighted supermatrix is acquired directly from pair-wise comparison ratios, (2) then the weighted supermatrix is calculated thereby multiplying the values by cluster weights and normalizing by column, and (3) the limited supermatrix provides the weights of the alternatives by converging the supermatrix (Saaty 2001).

## **27.4 Results**

### ***27.4.1 Survey Results***

In total, 111 responses were collected during the survey deployment period. It was discovered after selecting questions for the clustering analysis that due to omitted answers (none of the respondents were required to answer any particular question

**Table 27.1** Generalized characteristics of respondents—total group

Origin	Sex	Average age	About using and owning a Viper
USA, 80 %	F, 65 %	34.45	Average # of Vipers owned: 1.18
International, 20 %	M, 35 %		Average years owning Viper: 3.44

**Table 27.2** Generalized characteristics of respondents—total group

Level of playing	%	Instruments respondents also play		Music respondents play	
			%		%
Student	27	Only viper	07	Rock	85
Amateur	37	Violin or viola	80	Classical	82
Educator (Prim. income)	14	Mandolin	12	Bluegrass	27
Educator (Sec. income)	08	Guitar or bass	35	Country	23
Performer (Prim. income)	14	Cello	07	Jazz	50
Performer (Sec. income)	30	Piano	24	Celtic	34

except to indicate that they were in fact a Viper owner), some respondents had to be excluded from the analysis. This led to 74 valid respondents that we were able to include in this section of the methodology.

The results of the survey are shown in Table 27.1 where the descriptive statistics for each variable are displayed as well as in Table 27.2 where those descriptive statistics are converted to statements to describe the “typical Viper player.” While this information can be insightful in certain ways, in this particular format, it does not allow for many insightful design decisions to be made for a product family, so further analysis is required.

### 27.4.2 Cluster Analysis

The K-means clustering methodology was applied to the data collected from 74 respondents. The 25 original questions selected for the clustering were converted to 61 numeric variables, and the K-means methodology was applied to those variables. The scenarios for  $K = [2, 3, 4, 5, 6, 7, 8, 9, 10]$  were all completed and the results can be found in Table 27.3, while a more detailed set of results is provided for  $K = 5$  in Table 27.4.

As the methodology presented in Ray and Turi’s work (1999) suggests, the first local validity maximum was identified to be ( $K = 3$ , validity = 4.52) and the points after that [ $(K + 1, K\_max)$  or ( $K = [4, 10]$ )] were placed into consideration for the optimum number of product variants,  $N^*$ . One method of determining  $N^*$  is to select the  $K$  with the lowest validity measure within this range ( $K = [4, 10]$ ). This method would yield the choice of ( $K = 8$ , validity = 2.639), or  $N^* = 8$ .

**Table 27.3** K-values validity results

K # of clusters	Intra-cluster distance	Inter-cluster distance	Validity
2	55.940	18.934	2.955
3	54.010	11.947	4.521
4	52.913	13.272	3.987
5*	50.804	15.865	3.202
6	49.876	16.107	3.096
7	48.375	16.107	3.003
8#	46.636	17.674	2.639
9	45.413	15.103	3.007
10	44.361	15.103	2.937

\* Optimum with the highest bend

# Optimum with the lowest validity score

**Table 27.4** Detailed cluster analysis results for K = 5

K = 5	Number of Clusters observations	Within cluster	Avg. distance	Max. distance	Intra	Inter		
		sum of squares	from centroid	from centroid				
	Cluster 1 3	666.511	7.087	9.829	50.80393	15.86509		
	Cluster 2 1	1051.767	6.914	10.285				
	Cluster 3 2	530.598	6.630	7.241				
	Cluster 4 2	1138.805	7.089	10.65			MIN	3.9831
	Cluster 5 6	371.810	7.706	9.806				
Distance between cluster centroids		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5		
	Cluster 1	0.0000	5.2484	0.6386	4.0987	5.4187		
	Cluster 2	5.2484	0.0000	5.0016	4.9867	6.561		
	Cluster 3	4.6386	5.0016	0.0000	3.9831	5.6428		
	Cluster 4	4.0987	4.9867	3.9831	0.0000	5.2591		
	Cluster 5	5.4187	6.561	5.6428	5.2591	0.000		

An alternative method is to select the point with the largest “bend.” Bend is the improvement from one point to another, in the form of a decrease in validity. For example, the bend between points K = 3 and K = 4 with respective validity measures 4.521 and 3.987 is 0.534 (4.521 – 3.987 = 0.534). Although there is an improvement, or “bend” from K = 3 to K = 4, when examining other nearby points, it was determined that the bend from K = 4 to K = 5 was greater than the bend from K = 3 to K = 4. The point K = 5 has the largest bend of the group, so according to this method, K = 5 yields an N\* = 5.

According to the two approaches, both N\* = 5 and N\* = 8 are reasonable results to consider. When considering the two different scenarios, we can see the individual group membership is low when N\* = 8. Four out of the eight total groups have membership less than 7 %, and two of those groups have membership of less than 2 %. The % of the original sample that each cluster represents was also calculated

**Table 27.5** Group membership scenario  $N^* = 5$ 

Cluster (group) #	#Group members	% of Total group
1	13	17.57 %
2	21	28.38 %
3	12	16.22 %
4	22	29.73 %
5	6	08.11 %

for the  $K = 5$  ( $N^* = 5$ ) scenario. The findings are listed below in Table 27.5. The choice of  $N^* = 5$  has a more balanced membership than the  $N^* = 8$  scenario with only one group below 10 % membership. Based on this finding,  $N^* = 5$  was chosen over  $N^* = 8$  as the better suited final result for this particular situation. Also, as seen in Table 27.4, for  $K = 5$ , intra-cluster distance is 50.804, inter-cluster distance is 15.568, and validity is 3.202. These results are very reasonable.

Once  $N^*$  was determined through the clustering analysis, each group membership was identified and new descriptive statistics were calculated. With this information we can begin to identify their needs and preferences. Demographic profiles of these customer groups are provided in Table 27.6. Descriptions of each Viper player customer types #1–5 were generated from the new descriptive statistics and are shown in Table 27.7. The information in these tables was used to create concise lists of ranked customer requirements for QFD analysis. This information can also be used during many different stages of design and implementation such as the marketing and sales efforts.

### 27.4.3 Defining Customer Groups

Once the optimal number of variants in the product family ( $N^*$ ) was determined through the clustering analysis, membership in each group was identified and new descriptive statistics were calculated. With this information we are able to profile each group and identify their needs and preferences.

### 27.4.4 Implementing QFD: Fitting Designs to Customers

Customer needs for each group were extracted and ranked from the details of the new descriptive statistics described in the previous section. We can generate and rank the lists by observing the habits of each group. For instance, the typical “#3” type customer performs an average of over five instances per month, but the typical “#2” type customer makes less than one (if any) performance per month, so “ease of use during performance” appears on the needs list for “#3” but not “#2.” A list of the ranked customer requirements according to customer groups is provided in Table 27.8.

**Table 27.6** Detailed descriptions of customer groups

	Group #1	Group #2	Group #3	Group#4	Group #5
Average age	37.2	32.57	29.96	34.02	44.67
Sex	69.23 % male	66.67 % male	58.33 % male	68.18 % male	50 % male/50 % female
Avg. # of Viper owned	1.08	1.05	1.08	1.27	1.67
Avg. years of Viper owned	4.13 years	3.24 years	2.49 years	3.64 years	3.85 years
Avg. training level (1–6)	2.61	2.62	4.08	3.091	3.167
Avg. training level description	Between pre-college private lessons and adult private lessons	Between pre-college private lessons and adult private lessons	Near an undergraduate music degree	Near an undergraduate music minor	Between undergraduate music minor and music major

**Table 27.7** Overview of five Viper player groups

Group #1	Group #2	Group #3	Group #4	Group #5
Performers, amateurs, students	Amateurs, students, educators	Students, performers, educators	Mostly performers, some students, educators	Educators, performers, amateurs,
Rock, classical, country	Classical, rock, celtic, jazz	Always classical, very likely rock	Mostly rock, classical, all other styles	Classical, rock, jazz, bluegrass
Guitar, piano	Violin or viola	Always violin/viola, sometimes guitar/bass	Violin/viola, mandolin	Violin/viola, guitar/bass, piano, sings
Frequently practices multiple instruments at home	Sometimes practices multiple instruments at home	Spends the most time out of any group playing multiple instruments at home	Often plays multiple instruments at home	Plays multiple instruments at home
Frequent multi-instrumental performance	Very infrequent single instrument performer	Frequent multi-instrument performances but has infrequent intermissions	Frequent multi-instrument performances with frequent intermissions and switches	Frequent multi-instrument performer
Travels infrequently but with moderate equipment	Almost never travels, only with instrument	Travels recreationally with minimal equipment	Travels for fun and performing with multiple types heavy equipment	Infrequent, only for fun travel, minimal to moderate equipment

**Table 27.8** Ranked customer requirements (Groups #1–#5)

Requirements	Req. no	Group #1	Group #2	Group #3	Group #4	Group #5
Easy to use during performance	CR1	1		4	1	3
Ability to hold guitar	CR2	2		7		5
Cost	CR3	3	1	2	6	7
Stability of instrument	CR4	4	2	1	5	4
Aesthetics	CR5	5	3	3	7	6
Easy to retrieve instrument	CR6	6			2	1
Portability/local travel	CR7	7				
Accommodate violin/viola	CR8		4			
Air plane travel ready	CR9			5	4	
Can hold violin/viola	CR10			6	3	2

Possible concepts were generated for each part of the stand (Fig 27.3). This is an example of modular architecture which is discussed in the literature review. An instrument stand requires a top support (for the neck of the instrument), a rod (for the midsection or body of the stand), a bottom support (for the bottom of the

















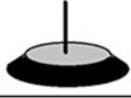
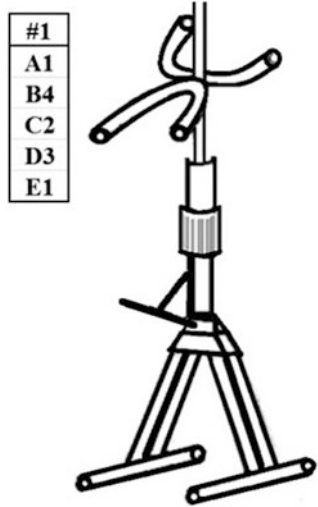
	CONCEPT	DESCRIPTION	DESIGN
TOP SUPPORT	#A1	U-GRIP HOLDER FOR NECK	
	#A2	U-GRIP HOLDER WITH ONE-ARM LOCK	
	#A3	PADDED U-GRIP HOLDER WITH TWO-ARM LOCK	
MULTI-INSTRUMENT CAPABILITY	#B1	HOLDS SINGLE VIPER	
	#B2	HOLDS VIPER AND VIOLIN, VIOLA, OR MANDOLIN	
	#B3	HOLDS TWO VIPERS	
	#B4	HOLDS VIPER AND GUITAR/BASS	
ROD TYPE	#C1	SINGLE HOLLOW ROD	
	#C2	TELESCOPING ROD	
BOTTOM SUPPORT	#D1	TWO BOTTOM GRIPS UNDER WINGS	
	#D2	ONE ROD BETWEEN WINGS	
	#D3	T-BAR ON BACK OF INSTRUMENT	
BASE	#E1	TWO T-BASE LEGS	
	#E2	TRIPOD BASE	
	#E3	SOLID DISC BASE	

Fig. 27.3 Generated concepts

instrument, in this case the wings of the Viper), and a base. This design problem also requires the decision of how many and what instruments the stand can hold, or “multi-instrument capability.” This defines whether the stand can accommodate one instrument or two instruments (in addition to a Viper, an acoustic violin or viola, a guitar, or an additional Viper). The type of bow holder was held constant and not considered as an option. These conceptual designs can be synthesized to form the best option for various customer groups (see Fig. 27.4, for an example).

Fig. 27.4 Final designs for Group #1



CUSTOMER #1 QFD MATRIX		COMPONENTS															
		Weight	#A1	#A2	#A3	#B1	#B2	#B3	#B4	#C1	#C2	#D1	#D2	#D3	#E1	#E2	#E3
CUSTOMER REQUIREMENTS	EASY TO USE IN PERFORMANCE	10	6	3						6			3		6	6	
	ABILITY TO HOLD GUITAR	9						9									
	COST	8	3			6				3		6					3
	STABILITY OF INSTRUMENT	7		9	6					3		9	6		3		6
	AESTHETICS	6	3							3				9	3		3
	EASY TO RETRIEVE INSTRUMENT	5	9	6										3			
	PORTABILITY FOR LOCAL TRAVEL	4								6							
<b>WEIGHTED SUM</b>			147	123	42	48	0	0	81	63	84	63	90	99	99	60	84
<b>% TOTAL</b>			0.14	0.11	0.04	0.04	0	0	0.07	0.06	0.08	0.06	0.08	0.09	0.09	0.06	0.08

Fig. 27.5 QFD matrix to fit designs to clusters

The QFD matrices that were used to fit designs to the customer groups are provided below in Fig. 27.5. The matrices are a combination of the typical QFD Phase 1 and QFD Phase 2, which allow us to directly link the customer requirements to concept design alternatives. One concept from each section

(A, B, C, D, E) has to be selected to make a final design. The customer requirements were pre-ranked and the calculation of the scores incorporated the ranking. The highest scoring concept from each section was selected for the final design (the highest scores are highlighted in bold).

The QFD analysis for concept selection resulted in five unique designs corresponding to each group. Each design meets different needs for each group. For example, the final design of CG #1 along with a component list is shown above in Fig. 27.4.

The final designs are as follows: customer group #1 features the capability to hold a Viper electric violin as well as a guitar or bass. The rod is telescoping and the legs are folding to make it to store in a smaller space. The t-bar supports the back of the Viper without obstructing the view of the instrument. Group #2 is the most stable of the designs, offering the instrument safety and security that these customers require. It features a solid base, two bottom supports, and a locking arm that secures the neck of the instrument in the stand. It also allows customers to hold an additional acoustic violin or viola. Customer group #3 also offers security of the instrument but allows for the stand to fold in order to save space. It only holds one instrument, the Viper, and has the locking arm over the front of the neck holder. The base can fold as well as the rod that supports the body of the Viper, making it possible to fold to save space for storage or transportation. Customer group #4 final design is for the performer who needs ease of retrieval and space saving for portability. The neck holder is simple, without the locking arm, to make it quick and easy to retrieve an instrument during a performance. It has a telescoping rod and a tripod style base, making it very easy to fold for portability and storage. It also has the capability to hold an acoustic violin as well as the Viper. The design for customer group #5 also addresses ease of retrieval, featuring the simple neck holder and the t-bar back support. However, it has increased stability by using the folding base instead of the tripod and a solid rod instead of a telescoping rod. It also has the ability to hold an acoustic violin as well as the Viper.

#### ***27.4.5 Consistency Verification of QFD Results with Analytical Network Process***

Analytic network process (ANP) is used to simultaneously fit generated design concepts to customer requirements and customer groups. In ANP model, main clusters and nodes are constructed first: customer groups (CG1, ..., CG5), customer requirements (CR1, ..., CR10), and design concepts (A, ..., E). Then, customer groups (CGs) are linked to customer requirements (CRs) and design concepts (DCs), and CRs are linked to DCs. DCs should satisfy both CGs and CRs. CRs should satisfy CGs. The ANP model is presented in Fig. 27.6.

Pair-wise comparisons were performed with Saaty's [23] 1–9 scale, where 1 represents equal importance and 9 represents extreme importance that favors

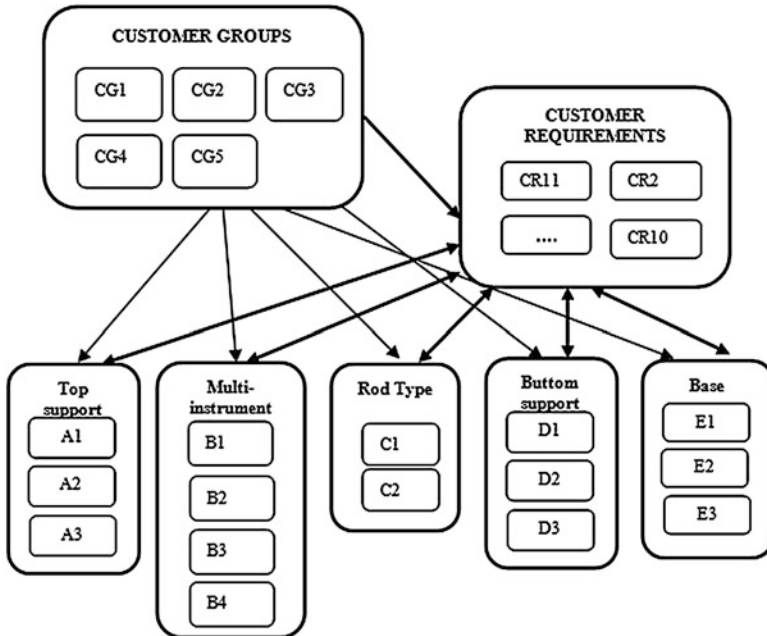


Fig. 27.6 ANP model of the study

one element over another. If the element has a weaker impact than its comparison element, then the scale ranges from 1 to 1/9 indicating indifference, or a relatively weak impact. The ANP model is solved using the software “Super Decisions” (<http://www.superdecisions.com>). First, all pair-wise comparison matrices are calculated and given in the form of unweighted supermatrix. A consistency test is conducted through the analysis of comparative matrix pairs using the consistency ratio (CR). CR is obtained through consistency index (CI) and random index (RI). When  $CR \leq 0.1$ , the comparative matrix pairs are deemed consistent. Then the weighted supermatrix (shown in Fig. 27.7) is transformed first to be stochastic. After entering the normalized values into the supermatrix, the supermatrix is then increased to a sufficient large power until convergence occurs. Figure 27.8 presents a final limit matrix. This limit matrix represents the final eigenvector.

The presented information in Fig. 27.7 shows us which concept is the most appropriate to meet customer requirements particular to each CG. For example, for CG1, most appropriate design concepts are A1, B4, C2, D3, and E1. A1 has highest priority (0.06063) between design concepts related top support for CG1 and B4 (0.09461) for multi-instrument capability, C2 (0.07596) for rod type, D3 (0.08782) for bottom support, and E1 (0.07797) for base. Note that the weighted supermatrix confirms the QFD-based design selection provided in Fig. 27.4.

Limit matrix shows that if we want to develop only one product that meets all CG’s requirements, we can determine which component is mostly suitable to requirements of all customer groups by using limit matrix results. A1 has highest

Cluster Node Labels	...	CG1	CG2	CG3	CG4	CG5...
TOP SUPPORT	A1	<b>0.06063</b>	0.03672	0.04527	0.06313	0.05991
	A2	0.05060	0.05508	0.05405	0.05187	0.05337
	A3	0.01730	0.03672	0.02921	0.01353	0.01525
MULTI INSTRUMENT	B1	0.06556	0.07967	0.06772	0.05681	0.03013
	B2	0.00968	0.08186	0.05696	0.10305	0.08360
	B3	0.00968	0.00900	0.00705	0.00983	0.00734
	B4	<b>0.09461</b>	0.00900	0.04779	0.00983	0.06654
ROD TYPE	C1	0.05712	0.11977	0.07135	0.03565	0.06654
	C2	<b>0.07596</b>	0.01331	0.06173	0.09743	0.06654
BUTTOM SUPPORT	D1	0.05590	0.08623	0.06878	0.06735	0.05579
	D2	0.07989	0.06071	0.09556	0.09698	0.07245
	D3	<b>0.08782</b>	0.07666	0.05927	0.05928	0.09537
BASE	E1	<b>0.07797</b>	0.06429	0.06791	0.07150	0.08522
	E2	0.04723	0.01098	0.05725	0.07922	0.04352
	E3	0.06610	0.11604	0.06615	0.04058	0.06256

Fig. 27.7 Weighted super matrix

Cluster Node Labels	...	CG1	CG2	CG3	CG4	CG5...
TOP SUPPORT	A1	<b>0.05150</b>	<b>0.05150</b>	<b>0.05150</b>	<b>0.05150</b>	<b>0.05150</b>
	A2	0.03691	0.03691	0.03691	0.03691	0.03691
	A3	0.02198	0.02198	0.02198	0.02198	0.02198
MULTI INSTRUMENT	B1	<b>0.02400</b>	<b>0.02400</b>	<b>0.02400</b>	<b>0.02400</b>	<b>0.02400</b>
	B2	0.02102	0.02102	0.02102	0.02102	0.02102
	B3	0.01877	0.01877	0.01877	0.01877	0.01877
	B4	0.02103	0.02103	0.02103	0.02103	0.02103
ROD TYPE	C1	0.03471	0.03471	0.03471	0.03471	0.03471
	C2	<b>0.03575</b>	<b>0.03575</b>	<b>0.03575</b>	<b>0.03575</b>	<b>0.03575</b>
BUTTOM SUPPORT	D1	0.03623	0.03623	0.03623	0.03623	0.03623
	D2	<b>0.04162</b>	<b>0.04162</b>	<b>0.04162</b>	<b>0.04162</b>	<b>0.04162</b>
	D3	0.03229	0.03229	0.03229	0.03229	0.03229
BASE	E1	0.03693	0.03693	0.03693	0.03693	0.03693
	E2	0.04110	0.04110	0.04110	0.04110	0.04110
	E3	<b>0.04618</b>	<b>0.04618</b>	<b>0.04618</b>	<b>0.04618</b>	<b>0.04618</b>

Fig. 27.8 A part of the limit matrix

priority (0.05150) between design concepts related top support for CG1 and B1 (0.02400) for multi-instrument capability, C2 (0.03575) for rod type, D2 (0.04162) for bottom support, and E3 (0.04618) for base as shown in Fig. 27.8. For one product meeting requirements of all customer groups, concepts can be determined as A1, B1, C2, D2, and E3. Although this may be one approach to the real-world business solution, given the variation in use habits of the customer groups, it is not possible to completely satisfy all customers with this one product.

## 27.5 Conclusions

As a result of the study, a family size of  $N^* = 5$  was decided for the Viper stand product family; five individual customer groups were defined with five distinct product variants mapped and assigned to each, where the consistency of decisions is also verified through ANP.

The proposed methodology provides a simple, reproducible way to make product family sizing decisions that can be customized to many different problem types. It can be effectively executed with basic statistical software by a team of designers with business and engineering backgrounds, without extensive knowledge of engineering design principles without sacrificing quality of work. The knowledge gained through the survey phase can be used not only in the methodology but throughout the design process in a multitude of ways. This research demonstrates the importance of product family size in the design process through the differences in each customer group and their requirements. Designing for one group would not accommodate the needs of all the customers even if it is based on “average” or “typical” values. Segmenting the customers also allows for further analysis in later stages such as specific pricing and targeted advertising by having greater customer insight for each product variant in the family. Overall, this methodology lays a solid foundation for a team beginning the design process and provides invaluable customer insights and information that can be used from the beginning of a project to the very end and beyond.

**Acknowledgments** This product family design project was introduced during the 2010 offering of the Design Decision Making (IE/EDSGN 549) course at Penn State University. Some of the designs included within this chapter are inspired by the designs generated by students enrolled in IE/EDSGN 549. We acknowledge their contributions.

## References

- Bhandare S, Allada V (2009) Scalable product family design: case study of axial piston pumps. *Int J Prod Res* 47:585–620
- Bicknell BA, Bicknell KD (1995) The road map to repeatable success—using QFD to implement change. CRC, Boca Raton, FL
- Chan L-K, Wu M-L (2002) Quality function deployment: a comprehensive review of its concepts and methods. *Qual Eng* 15(1):23–35
- Chiu M, Gupta S, Okudan GE (2008) A multi-stakeholder quality function deployment approach to support design decision-making. In: Industrial engineering research conference, Vancouver, CA
- Dahmus JB, Gonzalez-Zugasti JP, Otto KN (2000) Modular product architecture. *Des Stud* 22(5): 409–424
- Fujita K (2002) Product variety optimization under modular architecture. *Comput Aid Des* 34:953–965
- Fujita KF, Sakaguchi H, Akagi S (1999) Product variety deployment and its optimization under modular architecture and modules communalization. In: ASME design engineering technical conferences, Las Vegas, Nevada, 12–15 Sept 1999.

- Huang JJ, Tzeng GH, Ong CS (2005) Multidimensional data in multidimensional scaling using the analytic network process. *Pattern Recognit Lett* 26:755–67
- Huang GQ, Li L, Schulze L (2008) Genetic algorithm-based optimisation method for product family design with multi-level commonality. *J Eng Des* 19(5):401–416
- Jiao J, Zhang Y (2005) Product portfolio identification based on association rule mining. *Comput Aid Des* 37:149–172
- Jiao J, Tseng MM, Duffy VG, Lin F (1998) Product family modeling for mass customization. *Comput Ind Eng* 35(3–4):495–498
- Kumar D, Chen W, Simpson TW (2009) A market-driven approach to product family design. *Int J Prod Res* 47(1):71–104
- Li Z, Feng Y, Tan J, Wei Z (2008) A methodology to support product platform optimization using multi-objective evolutionary algorithms. *Trans Inst Measur Contr* 30:295–312
- MacQueen JB (1967) Some methods for classification and analysis multivariate observations. In: *Proceedings of the 5th symposium on math, statistics, and probability, Berkeley, CA*, pp 291–297
- Meade L, Sarkis J (1998) Strategic analysis of logistics and supply chain management systems using the analytical network process. *Logist Transport Rev* 34(3):201–215
- Meyer MH, Lehnerd AP (1997) *The power of product platforms*. The Free Press, New York
- Ray S, Turi RH (1999) Determination of the number of clusters in K-means clustering and application in color image segmentation. In: *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques, India, 2–3 (Sect. 3.2)*.
- Saaty TL (2001) Decision making with the ANP and the national missile defense example. In: *Proceedings of the 6th international symposium on the AHP, ISAHP 2001, Bern, Switzerland*, pp 365–382
- Salhieh SM (2007) A methodology to redesign heterogeneous product portfolios as homogeneous product families. *Comput Aid Des* 39:1065–1074
- Super Decisions Software, <http://www.superdecisions.com>. Viewed on 20 Jul 2012
- Tseng MM, Jiao J (1996) Design for mass customization. *Ann CIRP* 45(1):153–156
- Tucker CS, Kim HM (2009) Data-driven decision tree classification for product portfolio design optimization. *J Comput Inform Sci Eng* 9:1–14
- Wagstaff K, Rogers S, Schroedl S (2001) Constrained K-means clustering with background knowledge. In: *Proceedings of the 18th international conference on machine learning*, pp 577–584
- Zha XF, Sriram RD (2006) Platform-based product design and development: A knowledge-intensive support approach. *Knowl Based Syst* 19:524–543

# Chapter 28

## Product Family Design and Recovery for Lifecycle

Minjung Kwak and Harrison Kim

**Abstract** Product family design via component sharing is a widely practiced approach for offering sufficient variety to the market in an economical way. Most of the previous research has focused on the benefits of product family in the design and manufacturing stages—early stages of product family lifecycle. This chapter highlights another important aspect of product family design—the impact of component sharing on product end-of-life management—by a quantitative model for evaluating product family design from an end-of-life perspective. The model identifies an optimal strategy for managing product take-back and end-of-life recovery by use of mixed integer programming, thereby assessing the product family design in terms of its profitability in end-of-life management. A design study of a smartphone family is presented, and the results show that the model can assess profitability of a family design and highlight preferred family design alternatives at various degrees of component sharing.

---

The authors published the original version in *Engineering Optimization Journal*, Vol. 43, Issue 3, 2011 (DOI:[10.1080/0305215X.2010.482990](https://doi.org/10.1080/0305215X.2010.482990)), which was modified for this book chapter.

M. Kwak  
Department of Industrial and Information Systems Engineering,  
Soongsil University, Seoul, Korea  
e-mail: [mkwak@ssu.ac.kr](mailto:mkwak@ssu.ac.kr)

H. Kim (✉)  
Department of Industrial and Enterprise Systems Engineering,  
University of Illinois at Urbana-Champaign, Urbana, IL, USA  
e-mail: [hmkim@illinois.edu](mailto:hmkim@illinois.edu)



## 28.1 Introduction

For more than a decade, a great deal of research has been conducted on the design issues expressed in the following questions. Can a set of products benefit a company when it is designed to have common components? If so, what are the best designs for a group of products by sharing certain components? It is commonly accepted now that sharing components across multiple products can have a multitude of benefits, especially in the design and manufacturing stages. Specifically, component sharing is highlighted as a means of increasing product variety while retaining the necessary economies of scale and scope (Simpson et al. 2006). The growing interest in component sharing has triggered the development of product family design. Many approaches have been developed to support component sharing and product family design, and successful product families have been reported by both academics and industries.

Most existing methods and applications, however, have overlooked the impact of product family design on product end-of-life management. Managing end-of-life products involves two major activities, i.e., product take-back for collecting used products from their former users and end-of-life recovery of economic value. Environmental regulations currently mandate that manufacturers assume the economic burden of these two activities (Mangun and Thurston 2002); therefore, manufacturers must find a way to achieve profitability in end-of-life management. The point is that the profitability of end-of-life management may be influenced by the design of the product family.

End-of-life management involves multiple types of end-of-life products. Accordingly, product take-back and end-of-life recovery are influenced by individual product designs and the interactions between designs, i.e., the commonality of components across product variants. Manufacturers must carefully make commonality decisions in product family design to improve profitability of end-of-life management. Thus, a method is needed to determine which product family design is better from an end-of-life perspective.

This chapter presents a quantitative model for assessing the profitability of product family designs in end-of-life management. The proposed model evaluates a product family for which the product variants are assumed to overlap end-of-life stages. Each product variant has a hierarchical assembly structure, and some of its components can be shared with other product variants. The model focuses on the fact that component commonality influences the end-of-life profitability by increasing the degree of component interchangeability. The model also identifies an optimal strategy for maximizing the profitability of managing product take-back and end-of-life recovery, which is formulated as a mixed integer programming problem.

The rest of the chapter is organized as follows. The background for the chapter is presented in Sect. 28.2, focusing on the problem settings and the end-of-life management process. The mathematical model to assess product family design is proposed in Sect. 28.3, and an illustrative example is presented in Sect. 28.4. Closing remarks are presented in Sect. 28.5.

## 28.2 End-of-Life Management of a Family of Products

### 28.2.1 Definition and Benefit of Product Family in End-of-Life Management

A family of products is defined as a group of related products that share a product platform—a set of common design elements, processes, technologies, and other assets (Jiao et al. 2007; Simpson 2004). In this research, a product family is specifically defined as a group of products (1) that has common components shared by some or all of its product variants and (2) whose product variants are anticipated to have overlapping end-of-life stages; i.e., end-of-life management can be performed on multiple product variants simultaneously. Sharing the product platform can benefit both design (prelife) and recovery (end-of-life) stages with this definition.

Figure 28.1 depicts a family of products in which two variants exist and Component X is common. Each product variant has a hierarchical assembly structure consisting of three levels—*core*, *intermediate* (Inderfurth and Langella 2008), and *component*. A core refers to a used product that is intact. Disassembly separates a core into parts that are either intermediates or components. Here, the term “part” refers to any decomposable element of a product. Intermediate denotes nonatomic parts of a product at the middle level of product hierarchy, which are neither a core nor a component. Through another step of disassembly, intermediates can be separated into child components. Component indicates an atomic part at the lowest level, which cannot be disassembled any further (Krikke et al. 1998). The parent items of a component can be either intermediates or cores, depending on the product structure. Starting from components, child parts are reassembled into a parent part until a core is made. It should be noted that all product variants in this study are assumed to have three-level structures for simplicity.

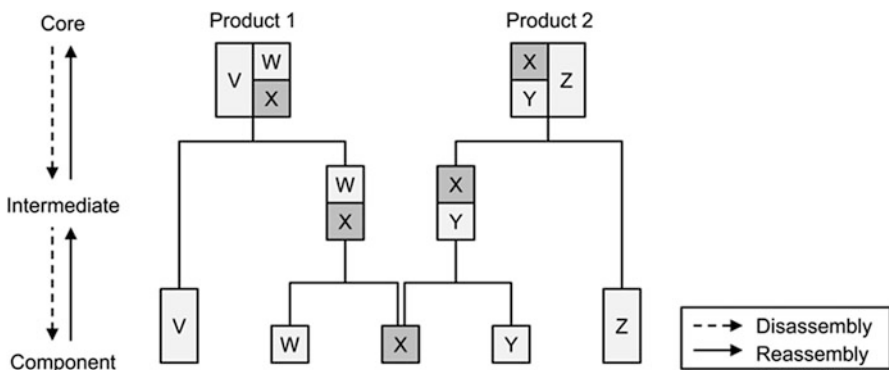


Fig. 28.1 Exemplary product family sharing Component X

With the definition of product family in the beginning of this section, product variants of a family of products have overlapping end-of-life stages. Hence, within a time period, multiple product variants are expected to reach the end-of-life stage at the same time, which renders component commonality across the variants and affects profitability in product end-of-life management. As Simpson (1998), Perera et al. (1999), and Bras (2007) stated, improving component commonality can benefit end-of-life management in two ways. First, the economies of scale in the recovery operation increase. Necessary tools and worker skill and setup time decrease in various operations, including disassembly, conditioning, and reassembly. Second, the interchangeability of components across a family of products increases. For instance, in Fig. 28.1, Component X, which resulted from the disassembly of Product 1, can be used for refurbishing both Intermediate WX and Product 1 and Intermediate XY and Product 2. Such increased interchangeability facilitates the profitable reuse of more components.

The current model in this chapter focuses on the increased interchangeability of components and its economic impact on the end-of-life stage, which has not been studied to any great extent in the previous literature. When a product family has some components that are shared by multiple variants, the model quantitatively assesses how the interchangeable components can increase manufacturer's profit. The developed model is one of the first attempts to examine product family design from the end-of-life point of view. Most previous approaches have focused on the initial manufacturing stage with aims to save product development efforts and maximize the profitability of new product sales. Their main concerns include what parts or design variables should be common among product variants and how the values of design variables should be optimized (e.g., Simpson et al. 2001; Fellini et al. 2005; Martin and Ishii 2002; Rai and Allada 2003; Simpson and D'Souza 2004; Alizon et al. 2009). Although a few studies (e.g., Simpson 1998; Perera et al. 1999; Bras 2007) considered the end-of-life stage, they simply state that cost reduction in the end-of-life stage is another possible advantage of component sharing.

## ***28.2.2 Processes for Product End-of-Life Management***

### **28.2.2.1 Product Take-Back**

This section describes the recovery processes under consideration in the model. End-of-life management consists of two sequential processes—product take-back and end-of-life recovery. Product take-back is the process of collecting cores, i.e., products that reach their end-of-life status. Since product take-back determines the volume, type, and quality of feedstock processed later in the recovery process, how

many cores and which types of cores should be acquired are major concerns for the manufacturer.

Regulatory requirements on waste collection greatly affect manufacturers' take-back decisions by forcing manufacturers to meet a certain collection target. For example, the Council of the European Union (EU) recently announced a new waste electrical and electronic equipment (WEEE) directive that imposes a mandatory collection target on EU member states (European Commission 2012). To comply with the legislation, member states must collect annually 45 % of the average weight for products positioned on their national markets. In this research, the proposed model assumes that a collection target exists for a manufacturing company. The company must take back a certain number of cores so that the total weight of the collected cores exceeds the target.

The cost of core procurement is another important factor that affects take-back decisions. According to environmental legislation, consumers can return the cores to collection points free of charge in most cases. Without compensation, however, consumers tend to store a core indefinitely even if they no longer use it (Kwak et al. 2011). Manufacturers provide an economic incentive to motivate consumers to return their cores. Although this may increase the take-back cost, manufacturers can secure a greater number of valuable cores in order to offset end-of-life management costs by making more profit in recovery. Thus, the proposed model assumes a buyback program as a take-back strategy. The buyback price can have either negative, zero, or positive value depending on the type and condition of the core. Negative value is included, because a company is allowed to charge consumers for taking back cores in some cases (Envirowise 2004).

For simplicity, the current study adopts bi-level condition levels, i.e., fully functioning (referred to as *working* hereafter) and malfunctioning (referred to as *nonworking* hereafter). Working cores are usually more expensive to buy back but have higher disassembly yield rates of working parts and components. Hence, the type, condition, and number of cores to take back should be carefully determined in end-of-life management.

### 28.2.2.2 End-of-Life Recovery

After product take-back, the collected cores pass through an end-of-life recovery process. Manufacturers must identify the most profitable way to recover incoming feedstock. To this end, this research considers recycling, reuse, reconditioning, refurbishment, and cannibalization as recovery options (Krikke et al. 1998; Jacobsson 2000; Kwak and Kim 2010). The meaning of each option is described in Table 28.1. Here, reuse and reconditioning options only apply to working items.

When deciding how to recover cores, manufacturers also need to consider environmental regulations, which obligate them to achieve a specific recovery rate or pay a penalty. In the proposed model, the recovery target is set at 80 % of

**Table 28.1** End-of-life recovery options

Option	Description
Recycling	An item is sent to recyclers and shredded, separated, and refined to recover raw materials. The higher the per weight material concentration, the more per weight recycling revenue
Reuse	An item is sold to another user to be used for its original purpose. Only essential operations (e.g., data scrubbing) are conducted without any value-adding operations. Only working cores can follow this option
Reconditioning	An item is sold to another user and used for its original purpose. In addition to essential operations, some minor value-adding operations, such as cleaning, lubricating, and polishing, are conducted to raise the value of the core. Only working cores can follow this option
Refurbishment	An item is restored to its original condition. Product type and structure are maintained. Disassembly, part conditioning and replacement, and reassembly belong to the refurbishment option. If upgrading functions are conducted to the level of up-to-date products, such refurbishment can be reclassified as remanufacturing
Cannibalization	An item is cannibalized for parts. Disassembly is conducted to separate a core into a set of parts. Individual parts resulting from the disassembly then can start their recovery as independent units, each with its own recovery and disposal option. Working parts can also be a source of parts for refurbishing other parts or cores

the collection target. For example, a company that has a collection target of 85,000 lb should recover more than 68,000 lb of resources from the collected cores. Many manufacturers (e.g., HP, Dell, and Apple) prefer complying with the regulation to paying penalties to promote “green” corporate image. Therefore, the proposed model represents the regulation as a constraint, which must be satisfied.

Figure 28.2 depicts the three-stage recovery process considered in this research. Here, a company is assumed not to carry out recycling operations on its own account. Instead, the company sells cores and parts to its recycling partners who perform actual recycling operations. Depending on the path each core follows in the recovery process, a set of collected cores can be transformed into eight kinds of outputs, i.e., four in the form of a product and four in the form of parts. These outputs are further transported to landfills, recycling partners, or customer markets, according to their assigned disposal and recovery options.

In the first stage of the recovery process, a decision is made concerning the next step for each core collected from the product take-back. To illustrate, suppose a set of used cell phones just arrived for the recovery process. Based on their conditions, the cell phones are discarded, recycled, reused, reconditioned, or disassembled. Cell phones for disposal and material recycling go to landfills or to recycling partners. The other cell phones undergo data scrubbing to eliminate any remaining personal data, and some of them are resold as reused or reconditioned phones, and some are sent to Stage 2 for disassembly.

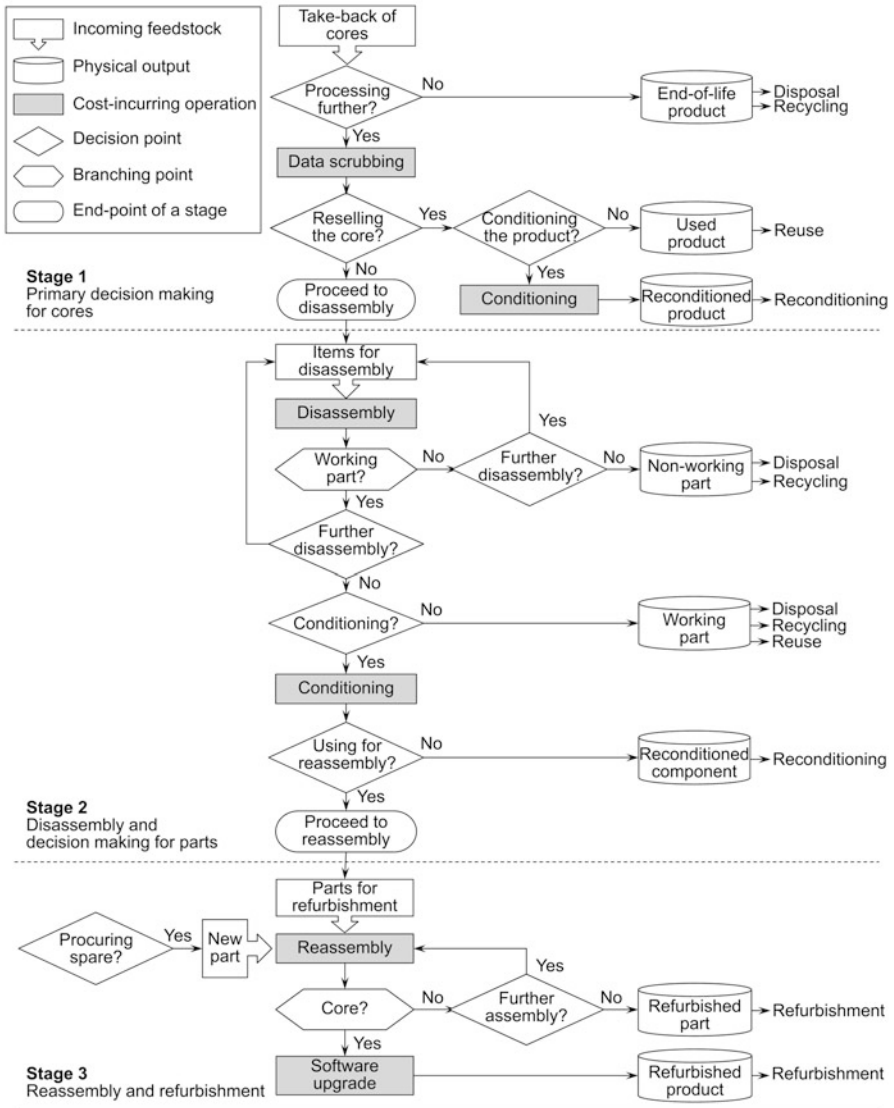


Fig. 28.2 End-of-life recovery process

In Stage 2, a core is disassembled for the purpose of refurbishment or cannibalization. For example, a cell phone from Stage 1 is disassembled into a screen module, main board, antenna, microphone, keypad, and cases. Further disassembly can be done as needed, but an important point is that every resultant part is either working or nonworking. A deterministic parameter, disassembly yield

rate, reflects the number of working parts acquired by the disassembly of a core or an intermediate. Similar to the approach taken by Krikke et al. (1998), disassembly yield rate in this research depends on the parent item's condition. For example, suppose a cell phone has the following disassembly yield rates for its main board:  $Yield|W = 1$  and  $Yield|N = 0.8$ . When a working (W) cell phone is disassembled, one unit of working main board results from the disassembly. When a nonworking (N) cell phone is disassembled, only 0.8 unit of working mother board is harvested. The remaining 0.2 unit is nonworking.

After disassembly, nonworking parts are either disposed of or recycled. For working parts, any disposal and recovery options are allowed including reassembly. If the reassembly option is chosen, a part is harvested, reconditioned, and sent to Stage 3. In Stage 3, parts are reassembled into its parent part or a core. When there is a shortage of parts, new spare parts are procured. The resulting parts and cores are remarketed as refurbished items.

## **28.3 Model for Evaluating Product Family Design from an End-of-Life Perspective**

### ***28.3.1 Problem Statement***

The proposed model uses mixed integer programming to identify an optimal take-back and recovery strategy for a given a product family design. The optimization result can be used to quantify the economic impacts of component sharing on end-of-life management. The proposed model is summarized by the following optimization problem:

- Given
  - Product family design with predefined commonality decisions.
  - Disassembly yield rates of cores and intermediates.
  - Costs of cores and the maximum amount of cores available for take-back.
  - Costs and revenue of executing recovery and disposal options.
  - Market demand for recovered items.
- Find
  - Optimal take-back strategy: amount, type, and condition of core that should be taken back.
  - Optimal disposal and recovery strategy: amount, type, and condition of core that should follow each disposal and recovery option; disassembly level of a core (parts to which a core should be disassembled) and recovery and disposal options for parts; amount and type of spare parts to acquire.

- Subject to
  - Flow volume balance constraints: With respect to an item, its flow balance between input and output units should be maintained.
  - Environmental regulations: Collection and recovery targets should be satisfied.
  - Core availability: There are limits on the amount of available cores that can be collected.
  - Avoiding excess fulfilment: The supply of a recovered item cannot exceed the demand for it.
- Maximizing total net profit from end-of-life management.
- Assuming
  - Three-level product structure: Each product variant has a three-level assembly structure consisting of a core, intermediates, and components, which are denoted with three indices.
  - Unlimited part procurement: Spare parts can be procured with no lead time, and there are no limits on the number of parts that can be purchased.
  - Unlimited facility capacity: There are no limits on the number of items or the number of operations that can be processed.
  - No loss in yield in the recovery operation: Data scrubbing, conditioning, disassembly, and reassembly do not damage their input items, and there is no loss in yield caused by operations.
  - Deterministic parameter values: Disassembly yield rates, market demand, related costs, and revenue are deterministic.
  - Single-period planning.

## 28.3.2 *Mathematical Formulation*

### 28.3.2.1 **Objective Function**

The objective of this model is to maximize the total profit from end-of-life management. The objective function is modeled in Eq. (28.1). The total cost of end-of-life management is the sum of nine cost components: (1) cost for take-back ( $C_1$ ), (2) cost for data scrubbing ( $C_2$ ), (3) cost for product conditioning ( $C_3$ ), (4) cost for disassembly ( $C_4$ ), (5) cost for part conditioning ( $C_5$ ), (6) cost for spare part procurement ( $C_6$ ), (7) cost for reassembly ( $C_7$ ), (8) cost for software update ( $C_8$ ), and (9) cost for disposal ( $C_9$ ). The total recovery revenue is the sum of four revenue terms: revenue from selling items to recyclers ( $R_1$ ), revenue from selling used items to the market ( $R_2$ ), revenue from selling reconditioned items to the market ( $R_3$ ), and revenue from selling refurbished items to the market ( $R_4$ ). The notations used in this chapter are described in Tables 28.2 and 28.3.



$$\begin{aligned}
\min : & \sum_{n=1}^9 C_n - \sum_{n=1}^4 R_n \\
\text{where} & \\
C_1 = & \sum_{i \in I} (c_{i,w}^t \cdot X_{i,w}^t + c_{i,n}^t \cdot X_{i,n}^t) \\
C_2 = & \sum_{i \in I} c_i^e \cdot (X_{i,w}^u + X_{i,w}^c + X_{i,w}^d + X_{i,n}^d) \\
C_3 = & \sum_{i \in I} c_i^c \cdot X_{i,w}^c \\
C_4 = & \sum_{i \in I} c_i^d \cdot (X_{i,w}^d + X_{i,n}^d) + \sum_{j \in J} c_j^d \cdot (X_{j,w}^d + X_{j,n}^d) \\
C_5 = & \sum_{j \in J} c_j^c \cdot (X_{j,w}^c + X_{j,w}^r) + \sum_{k \in K} c_k^c \cdot (X_{k,w}^c + X_{k,w}^r) \\
C_6 = & \sum_{j \in J} c_j^y \cdot Y_j + \sum_{k \in K} c_k^y \cdot Y_k \\
C_7 = & \sum_{i \in I} c_i^r \cdot Z_i^s + \sum_{j \in J} c_j^r \cdot (Z_j^r + Z_j^s) \\
C_8 = & \sum_{i \in I} c_i^s \cdot Z_i^s \\
C_9 = & \sum_{i \in I} c_i^l \cdot (X_{i,w}^l + X_{i,n}^l) + \sum_{j \in J} c_j^l \cdot (X_{j,w}^l + X_{j,n}^l) + \sum_{k \in K} c_k^l \cdot (X_{k,w}^l + X_{k,n}^l) \\
R_1 = & \sum_{i \in I} r_i^m \cdot (X_{i,w}^m + X_{i,n}^m) + \sum_{j \in J} r_j^m \cdot (X_{j,w}^m + X_{j,n}^m) + \sum_{k \in K} r_k^m \cdot (X_{k,w}^m + X_{k,n}^m) \\
R_2 = & \sum_{i \in I} r_i^u \cdot X_{i,w}^u + \sum_{j \in J} r_j^u \cdot X_{j,w}^u + \sum_{k \in K} r_k^u \cdot X_{k,w}^u \\
R_3 = & \sum_{i \in I} r_i^c \cdot X_{i,w}^c + \sum_{j \in J} r_j^c \cdot X_{j,w}^c + \sum_{k \in K} r_k^c \cdot X_{k,w}^c \\
R_4 = & \sum_{i \in I} r_i^z \cdot Z_i^s + \sum_{j \in J} r_j^z \cdot Z_j^s
\end{aligned} \tag{28.1}$$

### 28.3.2.2 Constraints

#### Flow Balance of Cores

There are several ways to process collected working cores, i.e., sending to landfills, selling to recyclers, selling as a used product, selling as a reconditioned product, and conducting disassembly to refurbish or cannibalize the core. For nonworking cores, the three available options are disposal, recycling, and disassembly. Constraint (28.2) requires every collected core to follow one of the possible options:

**Table 28.2** Mathematical notation (decision variable)

Notation	Description
<i>Index</i>	
$I, J, K$	Index set for core $i$ , intermediate $j$ , and component $k$ , respectively
$P_j, P_k$	Parent set of intermediate $j$ and parent set of component $k$ , respectively
$Q$	Quality condition set; $Q = \{w, n\}$ ; $q \in Q$
$w, n$	Working and nonworking quality condition index, respectively
<i>Variable</i>	
$X_{i,q}^l$	Number of core $i$ with condition $q$ to take back
$X_{i,q}^l, X_{j,q}^l, X_{k,q}^l$	Number of core $i$ , intermediate $j$ , and component $k$ with condition $q$ to dispose of, respectively
$X_{i,q}^m, X_{j,q}^m, X_{k,q}^m$	Number of core $i$ , intermediate $j$ , and component $k$ with condition $q$ to recycle, respectively
$X_{i,q}^u, X_{j,q}^u, X_{k,q}^u$	Number of core $i$ , intermediate $j$ , and component $k$ with condition $q$ to reuse, respectively
$X_{i,q}^c, X_{j,q}^c, X_{k,q}^c$	Number of core $i$ , intermediate $j$ , and component $k$ with condition $q$ to recondition, respectively
$X_{i,q}^d, X_{j,q}^d$	Number of core $i$ and intermediate $j$ with condition $q$ to disassemble, respectively
$X_{j,q}^r, X_{k,q}^r$	Number of intermediate $j$ and component $k$ with condition $q$ to use in refurbishment, respectively
$Y_j, Y_k$	Number of intermediate $j$ and component $k$ to procure for spare, respectively
$Z_j^r$	Number of intermediate $j$ to refurbish and use in core refurbishment
$Z_i^s, Z_j^s$	Number of core $i$ and intermediate $j$ to refurbish and sell in the market

$$\begin{aligned}
 X_{i,w}^t &= X_{i,w}^l + X_{i,w}^m + X_{i,w}^u + X_{i,w}^c + X_{i,w}^d \quad \forall i \in I \\
 X_{i,n}^t &= X_{i,n}^l + X_{i,n}^m + X_{i,n}^d \quad \forall i \in I
 \end{aligned}
 \tag{28.2}$$

### Flow Balance of Intermediates

Constraint (28.3) restrains the flow balance of working and nonworking intermediates, respectively. The left-hand side of each equation represents the amount of intermediates obtained from the disassembly of their parent cores. Since the model assumes a bi-level quality condition for every item, every intermediate acquired from the disassembly is either working or nonworking. Depending on the condition of the parent cores, the amount of working and nonworking intermediates can vary. To reflect this, the number of disassembled working cores and nonworking cores are multiplied by different disassembly yields  $\pi$ .

Each earned intermediate must follow one of the possible processing options. For working intermediates, six options are available: disposal, recycling, reuse, reconditioning, disassembly into components, and reuse for core refurbishment. For nonworking intermediates, only three options are available: disposal, recycling, and disassembly into components.

**Table 28.3** Mathematical notation (parameter)

Notation	Description
$\pi_{i,j}^o, \pi_{i,k}^o$	Number of units of intermediate $j$ and component $k$ that are originally included in core $i$ , respectively; the multiplicity of intermediate $j$ and component $k$
$\pi_{i,k}^o$	Number of units of component $k$ originally included in intermediate $j$
$\pi_{i,j}^q, \pi_{i,k}^q$	Disassembly yield rates of core $i$ with condition $q$ with respect to working intermediate $j$ and working component $k$ , respectively
$\pi_{j,k}^q$	Disassembly yield rates of intermediate $j$ with condition $q$ with respect to working component $k$
$\alpha, \beta$	Collection target and the maximum allowed disposal amount (recovery target)
$A_{i,q}$	Number of core $i$ with condition $q$ available for take-back
$\omega_i, \omega_j, \omega_k$	Weight of core $i$ , intermediate $j$ , and component $k$ , respectively
$c_{i,q}^t$	Unit take-back cost for core $i$ with condition $q$
$c_i^s$	Unit data scrubbing cost for core $i$
$c_i^c, c_j^c, c_k^c$	Unit conditioning cost for core $i$ , intermediate $j$ , and component $k$ , respectively
$c_i^d, c_j^d$	Unit disassembly cost for core $i$ and intermediate $j$ , respectively
$c_i^r, c_j^r$	Unit reassembly cost for core $i$ and intermediate $j$ , respectively
$c_j^c, c_k^c$	Unit procurement cost for intermediate $j$ and component $k$ , respectively
$c_i^s$	Unit software upgrade cost for core $i$
$c_i^l, c_j^l, c_k^l$	Unit disposal cost for core $i$ , intermediate $j$ , and component $k$ , respectively
$r_i^m, r_j^m, r_k^m$	Unit revenue from recycling core $i$ , intermediate $j$ , and component $k$ , respectively
$r_i^u, r_j^u, r_k^u$	Unit revenue from reusing core $i$ , intermediate $j$ , and component $k$ , respectively
$r_i^c, r_j^c, r_k^c$	Unit revenue from reconditioning core $i$ , intermediate $j$ , and component $k$ , respectively
$r_i^z, r_j^z$	Unit revenue from refurbishing core $i$ and intermediate $j$ , respectively
$D_i^u, D_j^u, D_k^u$	Demand for used core $i$ , intermediate $j$ , and component $k$ , respectively
$D_i^c, D_j^c, D_k^c$	Demand for reconditioned core $i$ , intermediate $j$ , and component $k$ , respectively
$D_i^z, D_j^z$	Demand for refurbished core $i$ and intermediate $j$ , respectively

$$\sum_{i \in P_j} (\pi_{i,j}^w \cdot X_{i,w}^d + \pi_{i,j}^n \cdot X_{i,n}^d) = X_{j,w}^l + X_{j,w}^m + X_{j,w}^u + X_{j,w}^c + X_{j,w}^d + X_{j,w}^r \quad \forall j \in J$$

$$\sum_{i \in P_j} \left( (\pi_{i,j}^o - \pi_{i,j}^w) \cdot X_{i,w}^d + (\pi_{i,j}^o - \pi_{i,j}^n) \cdot X_{i,n}^d \right) = X_{j,n}^l + X_{j,n}^m + X_{j,n}^d \quad \forall j \in J \quad (28.3)$$

**Flow Balance of Components**

Constraint (28.4) ensures the flow balance of working and nonworking components, respectively. The left-hand side of each equation represents the amount of components that resulted from disassembly. Both a core and an intermediate can be the parents of a component depending on the product family design. The conditions of parent items determine the amount of working and nonworking components.

The first equation states that every working component must follow one of five options: disposal, recycling, reuse, reconditioning, and reuse for intermediate refurbishment. The second equation requires every nonworking component to be landfilled or recycled. Since a component is the lowest-level part, the option of disassembly is not considered in both constraints.

$$\begin{aligned}
& \sum_{i \in P_k} (\pi_{i,k}^w \cdot X_{i,w}^d + \pi_{i,k}^n \cdot X_{i,n}^d) + \sum_{j \in P_k} (\pi_{j,k}^w \cdot X_{j,w}^d + \pi_{j,k}^n \cdot X_{j,n}^d) \\
& = X_{k,w}^l + X_{k,w}^m + X_{k,w}^u + X_{k,w}^c + X_{k,w}^r \quad \forall k \in K \\
& \sum_{i \in P_k} \left( (\pi_{i,k}^o - \pi_{i,k}^w) \cdot X_{i,w}^d + (\pi_{i,k}^o - \pi_{i,k}^n) \cdot X_{i,n}^d \right) \\
& + \sum_{j \in P_k} \left( (\pi_{j,k}^o - \pi_{j,k}^w) \cdot X_{j,w}^d + (\pi_{j,k}^o - \pi_{j,k}^n) \cdot X_{j,n}^d \right) \\
& = X_{k,n}^l + X_{k,n}^m \quad \forall k \in K
\end{aligned} \tag{28.4}$$

#### Flow Balance of Refurbished Intermediates

Intermediates can be refurbished by reassembling working components. Working components can result from the disassembly of cores or from external procurement. Once refurbished, intermediates can be sold on the market or reassembled with other parts to refurbish cores. Constraint (28.5) forces a balance of the flow between input components and output refurbished intermediates:

$$\sum_{j \in P_k} \pi_{j,k}^o \cdot (Z_j^r + Z_j^s) = X_{k,w}^r + Y_k \quad \forall k \in K \tag{28.5}$$

#### Flow Balance of Refurbished Cores

Similar to intermediates, cores can be refurbished by reassembling their working child parts. Working intermediates can be obtained by core disassembly or intermediate refurbishment. If there is a shortage of working intermediates, external procurement is also possible. As for the working components, only two sources are available: core disassembly and external procurement. After reassembly, refurbished cores are sold in the market as refurbished products. The first equation in Constraint (28.6) restricts the flow balance between input intermediates and output refurbished cores, while the second equation balances the flow between input components and output refurbished cores:

$$\begin{aligned} \sum_{i \in P_j} \pi_{i,j}^o \cdot Z_i^s &= X_{j,w}^r + Y_j + Z_j^r \quad \forall j \in J \\ \sum_{i \in P_k} \pi_{i,k}^o \cdot Z_i^s &= X_{k,w}^r + Y_k \quad \forall k \in K \end{aligned} \quad (28.6)$$

### Environmental Regulations

Environmental regulations require weight-based calculations. Constraint (28.7) represents the regulation on collection targets. The proposed model presumes a collection target  $\alpha$  for a manufacturing company. The company must take back enough cores to exceed the target.

$$\sum_{i \in I} \omega_i \cdot (X_{i,w}^t + X_{i,n}^t) \geq \alpha \quad (28.7)$$

Constraint (28.8) models the regulation on the minimum rate of recovery (or the maximum allowable disposal amount). The left-hand side of the constraint represents the total weight of discarded items, and  $\beta$  denotes the upper limit of disposal. In the proposed model, the recovery target is set at 80 % of the collection target  $\alpha$ . In other words, disposal of up to 20 % of  $\alpha$  is allowed; thus,  $\beta = 0.2\alpha$ .

$$\sum_{i \in I} \omega_i \cdot (X_{i,w}^l + X_{i,n}^l) + \sum_{j \in J} \omega_j \cdot (X_{j,w}^l + X_{j,n}^l) + \sum_{k \in K} \omega_k \cdot (X_{k,w}^l + X_{k,n}^l) \leq \beta \quad (28.8)$$

### Core Availability

The proposed model assumes a buyback program wherein the manufacturer pays the consumer for each core. The number and type of cores to take-back are decision variables, not given parameters. Regarding take-back decisions, Constraint (28.9) limits the amount of available cores that can be collected:

$$X_{i,w}^t \leq A_{i,w}; X_{i,n}^t \leq A_{i,n} \quad \forall i \in I \quad (28.9)$$

### Demand Satisfaction and Avoidance of Excess Fulfillment

The customer market demands a certain amount of used, conditioned, and refurbished items. Constraint (28.10) ensures that the supply of recovered cores, intermediates, and components cannot exceed the corresponding demand:

$$\begin{aligned}
X_{i,w}^u &\leq D_i^u; X_{i,w}^c \leq D_i^c; Z_i^s \leq D_i^z \quad \forall i \in I \\
X_{j,w}^u &\leq D_j^u; X_{j,w}^c \leq D_j^c; Z_j^s \leq D_j^z \quad \forall j \in J \\
X_{k,w}^u &\leq D_k^u; X_{k,w}^c \leq D_k^c \quad \forall k \in K
\end{aligned} \tag{28.10}$$

### Variable Condition

All decision variables in the model represent numbers of items. Due to disassembly yields, the amount of intermediates and components acquired from the disassembly might not be integers. To absorb the decimals, the amount of items sent to landfills and the amount sold to recyclers are set as real numbers. The others are constrained as integers. Constraint (28.11) restrains these variable conditions:

$$\begin{aligned}
X_{i,q}^t, X_{i,q}^u, X_{i,q}^c, X_{i,q}^d, Z_i^s &\geq 0 \text{ and integer}; X_{i,q}^l, X_{i,q}^m \geq 0 \quad \forall i \in I, \forall q \in Q \\
X_{j,q}^u, X_{j,q}^c, X_{j,q}^d, X_{j,q}^r, Y_j, Z_j^r, Z_j^s &\geq 0 \text{ and integer}; X_{j,q}^l, X_{j,q}^m \geq 0 \quad \forall j \in J, \forall q \in Q \\
X_{k,q}^u, X_{k,q}^c, X_{k,q}^r, Y_k &\geq 0 \text{ and integer}; X_{k,q}^l, X_{k,q}^m \geq 0 \quad \forall k \in K, \forall q \in Q
\end{aligned} \tag{28.11}$$

## 28.4 Illustrative Example

This section presents an illustrative example with a smartphone family to illustrate how to apply the proposed model and how it supports decision making in product family design.

### 28.4.1 Smartphone Family Design

#### 28.4.1.1 Design Scenario

Suppose that a smartphone manufacturer makes new products in the first period and uses cores to offer secondhand items along with new products in the next period. Until now, the company has customized the design of each phone to a specific market segment using uniquely designed components. However, since the company offers various types of phones to the market at the same time, parts proliferation due to the core variety (Bras 2007) has become one of the biggest obstacles to making profits in recovery. To address this issue, the company is considering designing a family of products in which some parts are shared by product variants. The design team has developed a design alternative for a product family. Now, they want to know whether the family design actually supports the recovery business and, if so,

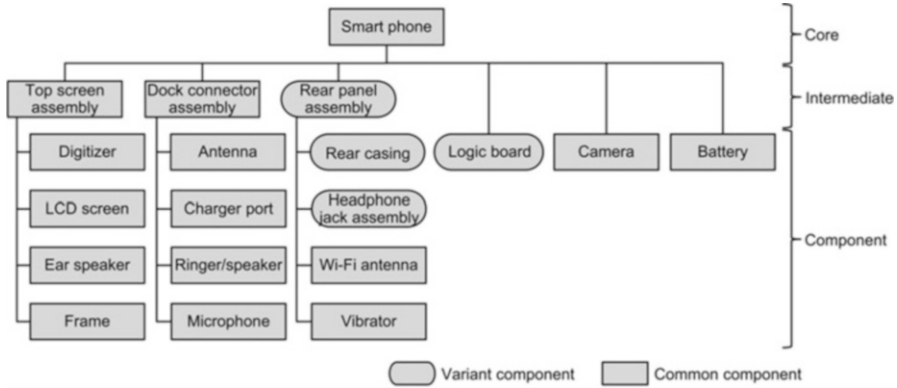


Fig. 28.3 Smartphone structure

what increase in profit is anticipated. Regarding the regulatory issues, the company currently has a collection target of 85,000 lb and a recovery target of 68,000 lb.

In this scenario, the proposed model is applied to a smartphone family (composed of four product variants). Figure 28.3 represents the product structure of the smartphone considered in this research. All product variants have identical structures composed of 15 components. The design difference comes from the variant parts, represented as an oval in the figure. Some, but not all, product variants can share the identical design for the variant parts. If all product variants share the same design for a part, the part is referred to as a common part (Thevenot and Simpson 2006). The smartphone variants differ in memory size and rear panel color.

Table 28.4 gives detailed information on the part composition of each product variant. Four product variants in this product family share a significant number of parts and intermediates noted as “common.” Numbers in Table 28.4 represents each type. For example, camera is noted as “1” for all four variants (i.e., common part), while rear casing is noted as 1, 2, 3, or 4, each representing different component.

For a simple illustration, the secondhand items to be recovered from cores are assumed to maintain their original design, without any hardware upgrade. In addition, the proposed model is applicable when the refurbished items have different design from cores; such refurbished items are regarded also as cores while their take-back availability ( $A_{i,q}$ ) is set as zero. By doing so, no take-back is considered for the second-generation products, but they become a possible throughput from refurbishment.

### 28.4.1.2 Parameter Setting

Table 28.5 represents the amount of available cores to take-back and the buyback price of a core for each type and condition. The parameter values used here are simulated based on the actual prices of a particular smartphone in the new product

**Table 28.4** Part composition of product variants in the high-sharing smartphone family

Part	Type of part	Type of commonality	Phone 1 (8 GB, black)	Phone 2 (16 GB, black)	Phone 3 (16 GB, white)	Phone 4 (32 GB, black)
Top screen assembly	Intermediate	Common	1	1	1	1
Dock connector assembly	Intermediate	Common	1	1	1	1
Rear panel assembly	Intermediate	Variant	1	2	3	4
Logic board	Component	Variant	1	2	2	3
Camera	Component	Common	1	1	1	1
Battery	Component	Common	1	1	1	1
Digitizer	Component	Common	1	1	1	1
LCD screen	Component	Common	1	1	1	1
Ear speaker	Component	Common	1	1	1	1
Frame	Component	Common	1	1	1	1
Antenna	Component	Common	1	1	1	1
Charger port	Component	Common	1	1	1	1
Ringer/speaker	Component	Common	1	1	1	1
Microphone	Component	Common	1	1	1	1
Rear casing	Component	Variant	1	2	3	4
Headphone jack assembly	Component	Variant	1	1	2	1
Wi-Fi antenna	Component	Common	1	1	1	1
Vibrator	Component	Common	1	1	1	1

**Table 28.5** Product take-back information

Type of core	Sale price (\$) (without 2-year contract)	$c_{i,w}^t$	$c_{i,n}^t$	$A_{i,w}$	$A_{i,n}$
Phone 1 (8 GB, black)	450	150	100	80,000	100,000
Phone 2 (16 GB, black)	550	180	100	50,000	80,000
Phone 3 (16 GB, white)	550	180	100	60,000	80,000
Phone 4 (32 GB, black)	650	300	150	10,000	10,000

market (<http://www.apple.com>), in the secondhand market (<http://www.ebay.com>), and in the buyback market (<http://www.gazelle.com>; <http://www.nextworth.com>). The price difference between cores originates mostly from the difference in memory size, which is determined by the logic board, the most expensive component in the smartphone family.

Table 28.6 shows the disassembly yield rates of cores and intermediates. In the smartphone family, most failures are expected in the top screen assemblies, especially the digitizers. The yield rates used in this study are estimated based on the failure reports on a particular smartphone model (SquareTrade 2008, 2009). The model has a



**Table 28.6** Disassembly yield rates

Parent part	Child part	Yield/W	Yield/N
Core	Top screen assembly	1	0.333
	Dock connector assembly	1	0.741
	Rear panel assembly	1	0.600
	Logic board	1	0.793
	Camera	1	0.787
	Battery	1	0.792
Top screen assembly	Digitizer	1	0.380
	LCD screen	1	0.545
	Ear speaker	1	0.718
	Frame	1	0.804
Dock connector assembly	Antenna	1	0.587
	Charger port	1	0.365
	Ringer/speaker	1	0.606
	Microphone	1	0.587
Rear panel assembly	Rear casing	1	0.407
	Headphone jack assembly	1	0.478
	Wi-Fi antenna	1	0.496
	Vibrator	1	0.496

structure similar to the one in Fig. 28.3. It should be also pointed out that, for simplicity, this study assumes the same yield rates for every core in the family. This is so for every rear panel assembly. If the parameter values are given, the proposed model can serve other cases as well. For example, the model is applicable to the case where different cores (e.g., Phone 1 and Phone 2) or different intermediates (e.g., rear panel assembly 1 and rear panel assembly 2) have different yield rates.

Finally, recovery cost and revenue parameters are assigned as shown in Tables 28.7 and 28.8. The disposal cost and recycling revenue of an item are assigned based on its weight shown in the second column. A cost per pound multiplier, \$0.02/lb (Sodhi and Reimer 2001), is used to estimate disposal costs. For recycling revenue, three different multipliers are used: \$5.00/lb for logic boards, \$1.50/lb for batteries, and \$2.50/lb for any mix of items (<http://www.grn.com>). Revenue from selling reused, reconditioned, and refurbished core is set as 30 %, 40 %, and 50 % of the new product price in Table 28.5. As for parts, the ratios change to 50 %, 65 %, and 80 % of new part price in the market. Retail prices of new parts are estimated according to the prices of similar parts in the market (<http://www.ubreakifix.com>). Note that these parameters can be changed for various market conditions. The proposed model can accommodate different set of parameters easily for different scenarios.

**Table 28.7** Recovery cost assumptions

Item	Weight (lb)	$c^e$	$c^c$	$c^d$	$c^r$	$c^y$	$c^s$	$c^l$
Phone 1	0.2908	1.5	1	1.5	2	–	1	0.0058
Phone 2	0.2952	1.5	1	1.5	2	–	1	0.0059
Phone 3	0.2952	1.5	1	1.5	2	–	1	0.0059
Phone 4	0.2996	1.5	1	1.5	2	–	1	0.0060
Front screen assembly	0.0960	–	0.5	0.5	1.5	56	–	0.0019
Lower dock assembly	0.0268	–	0.5	0.5	1.5	12	–	0.0005
Real panel assemblies	0.0672	–	0.75	0.5	1.5	38	–	0.0013
Logic board 1	0.0408	–	1	–	–	80	–	0.0008
Logic board 2	0.0452	–	1	–	–	100	–	0.0009
Logic board 3	0.0496	–	1	–	–	140	–	0.0010
Camera	0.0100	–	0.5	–	–	4	–	0.0002
Battery	0.0500	–	0.5	–	–	6	–	0.0010
Digitizer	0.0384	–	0.75	–	–	14	–	0.0008
LCD screen	0.0384	–	0.5	–	–	22	–	0.0008
Ear speaker	0.0096	–	0.5	–	–	4	–	0.0002
Mid chassis frame	0.0096	–	0.5	–	–	12	–	0.0002
Antenna	0.0038	–	0.5	–	–	4	–	0.0001
Dock connector	0.0096	–	0.5	–	–	6	–	0.0002
Loud speaker	0.0096	–	0.5	–	–	4	–	0.0002
Microphone	0.0038	–	0.5	–	–	3.6	–	0.0001
Rear casings	0.0384	–	0.5	–	–	30	–	0.0008
Headphone jack assemblies	0.0096	–	0.5	–	–	5.6	–	0.0002
Wi-Fi antenna	0.0096	–	0.5	–	–	4	–	0.0002
Vibrator	0.0096	–	0.5	–	–	4	–	0.0002

### 28.4.2 Optimization Result

In order to assess how much profit can be improved by adopting the family design, a reference case without component sharing was necessary. Therefore, a set of four smartphones that share no components (reference case in column 5 in Table 28.9) was analyzed in addition to the high-sharing family design (Family 1) described in Table 28.4, using equivalent parameters and assumptions. In addition, two families of smartphones with limited sharing are also derived and compared to examine how the degree of sharing influences the optimization result. One family (Family 2) shares only the digitizer and LCD screen across all product variants, and the other one (Family 3) shares the microphone only. To solve mixed integer programming, Risk Solver Platform and XPress Solver Engine were used.

The optimization results from the four different cases are shown in Table 28.9. Table 28.10 shows the optimal amount of cores to take back in each of the four cases. Due to the limitation of space, a complete set of optimization results is presented only for the high-sharing design (Family 1) in Table 28.11. Figure 28.4 is presented to help in understanding Table 28.11. It graphically represents some of the results in the table, specifically the results related to all cores, the front screen assembly, and

**Table 28.8** Recovery revenue and demand assumptions

Item	$r^m$	$r^u$	$r^c$	$r^z$	$D^u$	$D^c$	$D^z$
Phone 1	0.7270	135	180	225	10,000	10,000	10,000
Phone 2	0.7380	165	220	275	10,000	10,000	20,000
Phone 3	0.7380	165	220	275	10,000	10,000	20,000
Phone 4	0.7490	195	260	325	10,000	10,000	50,000
Front screen assembly	0.2400	70	91	112	20,000	20,000	40,000
Lower dock assembly	0.0670	15	19.5	24	20,000	20,000	20,000
Real panel assembly 1	0.1680	47.5	61.75	76	5,000	5,000	20,000
Real panel assembly 2	0.1680	47.5	61.75	76	5,000	5,000	15,000
Real panel assembly 3	0.1680	47.5	61.75	76	5,000	5,000	15,000
Real panel assembly 4	0.1680	47.5	61.75	76	5,000	5,000	10,000
Logic board 1	0.2040	100	130	–	5,000	5,000	–
Logic board 2	0.2260	125	162.5	–	10,000	10,000	–
Logic board 3	0.2480	175	227.5	–	5,000	5,000	–
Camera	0.0250	5	6.5	–	20,000	20,000	–
Battery	0.0750	7.5	9.75	–	20,000	20,000	–
Digitizer	0.0960	17.5	22.75	–	20,000	20,000	–
LCD screen	0.0960	27.5	35.75	–	20,000	20,000	–
Ear speaker	0.0240	5	6.5	–	20,000	20,000	–
Mid chassis frame	0.0240	15	19.5	–	20,000	20,000	–
Antenna	0.0095	5	6.5	–	20,000	20,000	–
Dock connector	0.0240	7.5	9.75	–	20,000	20,000	–
Loud speaker	0.0240	5	6.5	–	20,000	20,000	–
Microphone	0.0095	4.5	5.85	–	20,000	20,000	–
Rear casings	0.0960	37.5	48.75	–	5,000	5,000	–
Headphone jack assembly 1	0.0240	7	9.1	–	15,000	15,000	–
Headphone jack assembly 2	0.0240	7	9.1	–	5,000	5,000	–
Wi-Fi antenna	0.0240	5	6.5	–	20,000	20,000	–
Vibrator	0.0240	5	6.5	–	20,000	20,000	–

Note: When applying these parameters to a reference case, demand parameters (i.e.,  $D^u$ ,  $D^c$ , and  $D^z$ ) for shared parts are needed to change. For a product variant, the demand for each part is changed into  $(D/n)$ , where  $n$  is the number of variants sharing the part in the family design. For example, the demand for used camera is 5,000 (=20,000/4) for each phone in the reference case

the components that compose the front screen assembly. In the figure, a square corresponds to a specific part, and an arrow represents material flows in recovery processes with respect to the part. The type of an arrow indicates whether the flow is of working parts or of nonworking parts. The number and capital letter next to an arrow indicate the number of units of parts that follows a specific recovery or disposal option. For instance, the square “front screen assembly” and its relevant arrows in Fig. 28.4 show that the disassembly of Family 1 will provide 75,550 (working) and 146,600 (nonworking) units of front screen assembly. Out of the total 75,550 working parts, 20,000 units should be directly remarketed as used parts, another 20,000 units should be reconditioned and remarketed as reconditioned parts, and 35,500 units should be reused for core refurbishment. Recycling should be

**Table 28.9** Optimization result (objective value)

	Family 1 (high-sharing design)	Family 2 (sharing display only)	Family 3 (sharing MIC only)	Reference (no sharing)
Cost in total	50,193,370	50,413,117	50,454,510	50,411,329
Take-back	35,428,200	35,590,450	35,590,420	35,590,420
Data scrubbing	433,395	433,358	433,358	433,358
Core conditioning	36,804	30,000	30,000	30,000
Disassembly	516,793	519,440	519,976	519,441
Part conditioning	694,243	614,578	611,375	616,448
New part procurement	12,507,263	12,652,324	12,696,412	12,648,694
Reassembly	476,674	472,969	472,970	472,970
Software upgrade	100,000	100,000	100,000	100,000
Disposal	0	0	0	0
Revenue in total	70,494,557	68,944,233	68,843,196	68,788,606
Recycling	64,017	74,574	75,056	75,104
Reuse	12,580,000	12,788,185	12,686,640	12,632,003
Reconditioning	18,830,540	17,061,474	17,061,500	17,061,500
Refurbishment	39,020,000	39,020,000	39,020,000	39,020,000
Objective value (cost-revenue)	-20,301,186	-18,531,117	-18,388,687	-18,377,277
ROI (return on investment)	40.45 %	36.76 %	36.45 %	36.45 %

**Table 28.10** Optimal number of units of core  $[X_{i,w}^t, X_{i,n}^t]$  to take back

	Family 1	Family 2	Family 3	Reference
Phone 1 (8 GB, black)	[20000, 65548]	[19999, 64700]	[20000, 64701]	[20000, 64701]
Phone 2 (16 GB, black)	[20000, 72103]	[20000, 72103]	[19999, 72101]	[19999, 72101]
Phone 3 (16 GB, white)	[20000, 72103]	[20000, 72103]	[20000, 72104]	[20000, 72104]
Phone 4 (32 GB, black)	[9176, 10000]	[10000, 10000]	[10000, 10000]	[10000, 10000]
Total weight of cores to take back (lb)	[20373, 64627]	[20620, 64380]	[20620, 64380]	[20620, 64380]

chosen only for a small fraction of working and nonworking parts. For most of the nonworking units, further disassembly is required.

From the optimization results, three implications are obtained as follows:

- Result 1: Family 1 (high-sharing design) is the most profitable design among the four cases. In Table 28.9, all four cases present negative objective values, which implies that the end-of-life management can be a profitable business in all cases. Family 1 shows the smallest objective value (i.e., the largest profit). This means that Family 1 can support end-of-life management, and, once adopted, the profit is expected to increase by 1.9 million dollars.
- Result 2: Family 1 allows the most efficient end-of-life management among the four cases. The last row of Table 28.9 presents the return on investment (ROI) of each case. ROI denotes the ratio of net profit relative to the cost, and the higher, the better. The ROI for Family 1 is the highest, which means that Family 1 can obtain more profit for the same amount of investment.
- Result 3: Maximum profit and ROI increase as the degree of component sharing increases. Family 1 has the highest degree of component sharing, while the reference case has no sharing. Families 2 and 3 are in between these two. The four cases demonstrate that the maximum profit and ROI increase with the degree of component sharing. The results also indicate that the identity of the components that are shared is also an important factor affecting the profitability. For example, even though the microphone is shared in Family 3, the profit and ROI do not change much. Microphone and its parent intermediate (i.e., dock connector assembly) are the cheapest parts in a smartphone; although Family 3 encourages reuse of these parts (rather than material recovery), the revenue from the increased reuse is too small to make any significant difference in total profit. However, when the digitizer and LCD screen are shared in Family 2, the profit increases more significantly. Digitizers and LCD screens are the most high-priced components along with logic boards.

The discussion up to now has been focused only on the economic perspective. Table 28.12 interprets the same optimization results from a different viewpoint, i.e., material flows. Comparing the four cases gives the following implications:

- Result 4: Family 1 requires less new resources to retrieve maximum profit from the same amount of input material. The table shows how much material must be input to the recovery system to obtain maximum profit. Family 1 shows

**Table 28.11** Optimal solution to the end-of-life management of a smartphone family

Item	$X^t_w$	$X^t_n$	$X^m_w$	$X^u_w$	$X^c_w$	$X^d_w$	$X^r_w$	$X^m_n$	$X^d_n$	Y	$Z^r$	$Z^s$
Phone 1	20,000	65,548	0	10,000	10,000	0	0	0	65,548			10,000
Phone 2	20,000	72,103	0	10,000	10,000	0	0	0	72,103			20,000
Phone 3	20,000	72,103	0	10,000	10,000	0	0	0	72,103			20,000
Phone 4	9,176	10,000	0	0	6,804	2,372	0	0	10,000			50,000
Front screen assembly	75,550.1	146,575.9	0.1	20,000	20,000	0	35,550	0.9	146,575	1	64,449	40,000
Lower dock assembly	165,209.7	56,916.3	0.7	20,000	20,000	26,591	98,618	0.3	56,916	1,382	0	20,000
Real panel assembly 1	39,328.8	26,219.2	0.8	5,000	5,000	19,328	10,000	0.2	26,219	0	0	20,000
Real panel assembly 2	43,261.8	28,841.2	0.8	5,000	5,000	13,262	19,999	0.2	28,841	1	0	15,000
Real panel assembly 3	43,261.8	28,841.2	0.8	5,000	5,000	13,262	19,999	0.2	28,841	1	0	15,000
Real panel assembly 4	8,372.0	4,000.0	0.0	0	5,000	3,372	0	0.0	4,000	50,000	0	10,000
Logic board 1	51,979.6	13,568.4	31,979.6	5,000	5,000		10,000	13,568.4		0		
Logic board 2	114,355.4	29,850.6	54,355.4	10,000	10,000		40,000	29,850.6		0		
Logic board 3	10,302.0	2,070.0	0.0	5,000	5,000		302	2,070.0		49,698		
Camera	175,318.4	46,807.6	35,318.4	20,000	20,000		100,000	46,807.6		0		
Battery	176,417.2	45,708.8	36,417.2	20,000	20,000		100,000	45,708.8		0		
Digitizer	55,698.5	90,876.5	0.5	20,000	20,000		15,698	90,876.5		88,751		
LCD screen	79,883.4	66,691.6	0.4	20,000	20,000		39,883	66,691.6		64,566		
Ear speaker	105,240.9	41,334.2	0.9	20,000	20,000		65,240	41,334.2		39,209		
Frame	117,846.3	28,728.7	0.3	20,000	20,000		77,846	28,728.7		26,603		
Antenna	60,000.7	23,506.3	0.7	20,000	20,000		20,000	23,506.3		0		
Dock connector	47,365.3	36,141.7	0.3	20,000	20,000		7,365	36,141.7		12,635		
Loud speaker	61,082.1	22,424.9	1,082.1	20,000	20,000		20,000	22,424.9		0		
Microphone	60,000.7	23,506.3	0.7	20,000	20,000		20,000	23,506.3		0		
Rear casing 1	29,999.1	15,547.9	0.1	5,000	5,000		19,999	15,547.9		1		
Rear casing 2	25,000.3	17,102.7	0.3	5,000	5,000		15,000	17,102.7		0		
Rear casing 3	25,000.3	17,102.7	0.3	5,000	5,000		15,000	17,102.7		0		
Rear casing 4	5,000.0	2,372.0	0.0	0	5,000		0	2,372.0		10,000		

(continued)

**Table 28.11** (continued)

Item	$X_w^l$	$X_n^l$	$X_w^m$	$X_w^u$	$X_w^c$	$X_w^{cl}$	$X_w^r$	$X_n^m$	$X_n^{cl}$	Y	$Z^s$
Headphone jack assembly 1	64,192.7	<u>30,829.3</u>	0.7	15,000	15,000		34,192	30,829.3		10,808	
Headphone jack assembly 2	27,048.0	15,055.0	2.048.0	5,000	5,000		15,000	15,055.0		0	
Wi-Fi antenna	<u>92,822.9</u>	<u>44,302.1</u>	0.9	20,000	20,000		52,822	44,302.1		7,178	
Vibrator	<u>92,822.9</u>	<u>44,302.1</u>	0.9	20,000	20,000		52,822	44,302.1		7,178	

Note: Underlined numbers indicate the number of working and nonworking items obtained from parents' disassembly; all  $X_w^l$  and  $X_n^l$  are zero

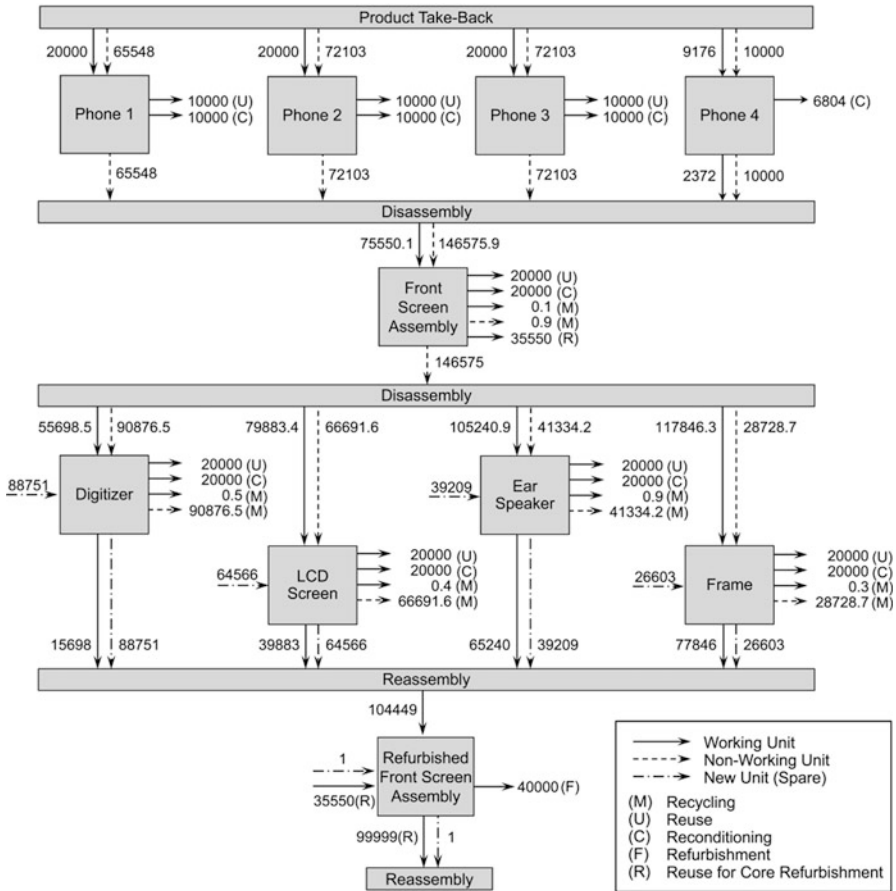


Fig. 28.4 Graphical representation of the part of the optimal solution

Table 28.12 Material input–output flow

	Family 1	Family 2	Family 3	Reference
Input (lb)				
Take-back	85,000	85,000	85,000	85,000
New part spare	13,128	16,529	16,545	16,454
Output (lb)				
Disposal	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)	0 (0.00 %)
Recycling	21,482 (21.89 %)	26,717 (26.31 %)	26,910 (26.50 %)	26,983 (26.58 %)
Reuse	17,988 (18.33 %)	18,192 (17.92 %)	18,016 (17.74 %)	17,916 (17.65 %)
Reconditioning	20,554 (20.95 %)	18,516 (18.24 %)	18,516 (18.23 %)	18,516 (18.24 %)
Refurbishment	38,104 (38.83 %)	38,104 (37.53 %)	38,104 (37.52 %)	38,104 (37.53 %)
Sum of weight (lb)	98,128	101,529	101,545	101,454
Net profit per pound (\$)	206.88	182.52	181.09	181.12



superiority here as well with use of smaller amount of new resources. From an environmental perspective, less new material is usually more desirable. However, it is hard to conclude that the higher degree of sharing is always better in terms of saving resources. Families 2 and 3 require more weight of material than the reference case. (Also, Family 3 is worse than the reference, even in the net profit per unit weight.) This is due to higher reuse rate of parts. In other words, more reusable parts are available due to higher interchangeability in product family compared to the reference case. In turn, the remanufacturer may use more material to manufacture more secondhand products for higher profit.

- **Result 5:** Family 1 supports end-of-life management to be more effective. First, Family 1 enables core management in a better way. From the environmental standpoint, reuse, reconditioning, and refurbishment are regarded as better options than material recycling. In this regard, Family 1 is superior to the others. Table 28.12 shows how the input material is processed in each design case. For Family 1, a greater percentage of material is reused, reconditioned, and refurbished than for the other cases. Second, Family 1 allows the retrieval of a greater value from the same amount of material. The last row of Table 28.12 shows net profit per pound. Family 1 shows the best performance in this area as well.

Family 1's superiority is the result of high interchangeability of its components. Thus, Family 1 can reduce take-back and part procurement costs while increasing recovery revenue. As shown in Table 28.5, Phone 4, which has 32 GB of memory and is black in color, is the most preferred phone to refurbish. The unit revenue obtained from refurbishment is higher than it is for other variants. In addition, the current setting of parameters assumes a large market demand for a refurbished Phone 4. (The demand for every core and part is listed in Table 28.8). However, Phone 4 is also the most difficult one to refurbish. Not only is it expensive to take-back, but the core availability is too low to satisfy the demand. While the demand for refurbished Phone 4 is 50,000, there are only 20,000 cores available, including working and nonworking cores. Therefore, spare parts must be purchased to meet the demand.

In the case of Family 1, a company can utilize other phones to refurbish Phone 4. Because Phone 1 is the least expensive core, it would be an excellent substitute for Phone 4. Accordingly, as presented in Table 28.10, the optimal take-back plan for Family 1 involves less take-back of working Phone 4 along with more take-back of nonworking Phone 1.

In addition to the cost reduction in product take-back and parts procurement, increased revenue for reconditioning is also examined. Since other phones take the place of Phone 4 in providing parts for refurbishment, the company can keep some of the available Phone 4 for other purposes without sacrificing refurbishment. Specifically, the company can recover Phone 4 by the second most profitable way; i.e., reconditioning, which increases the overall recovery revenue.

## 28.5 Summary

End-of-life management is regarded as a problem of multiple cores with commonality. In order to improve the profitability of end-of-life management, a manufacturer should make commonality decisions in product family design by considering their influences on product take-back and end-of-life recovery. To help manufacturers make the best decisions, an optimization method was developed for assessing product family designs for their profitability in end-of-life management. Using mixed integer programming, the model identifies an optimal strategy for product take-back and end-of-life recovery, thereby assessing the maximum profits for the product family during the end-of-life stage. The profit value can be used as a quantitative measure to evaluate product family design. By applying this method in the design stage, manufacturers can assess various product family designs and choose the best one for lifecycle.

An example with a smartphone family illustrates how to apply the proposed model and how it supports decision making in product family design. The study results demonstrate that product family design can be a means of improving the profitability of end-of-life management. When multiple products are designed to have common components, the profit outweighs that of the reference case with no components shared. Moreover, the superiority is examined not only in the magnitude of profit but also in the return on investment. The results also imply that the profit monotonically increases with the level of component sharing, but the increasing amount differs from case to case, depending on the shared parts. Finally, the results show that product family design has potential to support a more environmentally conscious product recovery. The high-sharing smartphone family produces greater value for the company from the same amount of material. Also, it requires a smaller amount of new material to achieve the maximum profit.

The economies of scale in recovery operations can be incorporated in the model in the future. As component commonality increases, the necessary tools, the required worker skills, and the time required for setup can decrease in various recovery operations (e.g., disassembly, conditioning, part purchasing, warehousing, and reassembly). However, in this paper, the economies of scale are excluded from consideration by assuming unlimited facility capacity and by assuming constant unit cost for every operation. Uncertainty is also an important aspect because many parameters, which are assumed to be known and deterministic in this chapter, are stochastic in reality. The developed model is one of the first attempts to examine product family design from the recovery point of view—thus using mixed integer programming was a natural choice—it is simple and provides a great foundation for a variety of studies in the future. However, uncertainty in real-world decisions must be considered in the assessment to find an optimal family design that is robust in handling possible changes. Future work should include the development of a stochastic model that can deal effectively with such uncertainties. Models for estimating the economic value of parts and the costs of recovery processes also need to be developed. Finally, an integrated approach should be developed in the

future that considers both end-of-life stage and design and manufacturing stages. Combining the proposed model with traditional family design approaches will lead to a more advanced framework.

**Acknowledgments** This material is based upon the work supported by the National Science Foundation under Award No. 0726934. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

- Alizon F, Shooter SB, Simpson TW (2009) Assessing and improving commonality and diversity within a product family. *Res Eng Des* 20(4):241–253
- Bras B (2007) Design for remanufacturing processes. In: Kutz M (ed) *Environmentally conscious mechanical design*. Wiley, Hoboken, NJ, pp 283–318
- Envirowise (2004) Sustainable design of electrical and electronic products to control costs and comply with legislation-GG427. Envirowise. <http://www.envirowise.gov.uk>. Accessed 30 Nov 2009
- European Commission (2012) Council steps up collection and recycling targets for wastelectrical and electronic equipment. <http://ec.europa.eu/environment/waste/weee>. Accessed 13 Jul 2012
- Fellini R, Kokkolaras M, Papalambros P, Perez-Duarte A (2005) Platform selection under performance bounds in optimal design of product families. *J Mech Des* 127(4):524–535
- Inderfurth K, Langella IM (2008) Planning disassembly for remanufacture-to-order systems. In: Gupta SM, Lambert AJD (eds) *Environment conscious manufacturing*. CRC, Boca Raton, FL, pp 387–411
- Jacobsson N (2000) Emerging product strategies – selling services of remanufactured products. Licentiate Dissertation, Lund University
- Jiao J, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18(1):5–29
- Krikke HR, van Harten A, Schuur PC (1998) On a medium term product recovery and disposal strategy for durable assembly products. *Int J Prod Res* 36(1):111–139
- Kwak M, Kim H (2010) Evaluating end-of-life recovery profit by a simultaneous consideration of product design and recovery network design. *J Mech Des* 132(7):071001
- Kwak M, Behdad S, Zhao Y, Kim H, Thurston D (2011) E-waste stream analysis and design implications. *J Mech Des* 133(10):101003
- Mangun D, Thurston D (2002) Incorporating component reuse, remanufacture, and recycle into product portfolio design. *IEEE Trans Eng Manag* 49(4):479–490
- Martin M, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. *Res Eng Des* 13(4):213–235
- Perera HS, Nagarur N, Tabucanon MT (1999) Component part standardization: a way to reduce the life-cycle costs of products. *Int J Prod Econ* 60–61:109–116
- Rai R, Allada V (2003) Modular product family design: agent-based pareto-optimization and quality loss function-based post-optimal analysis. *Int J Prod Res* 41(17):4075–4098
- Simpson TW (1998) A concept exploration method for product family design. Dissertation, Georgia Institute of Technology
- Simpson TW (2004) Product platform design and customization: status and promise. *Artif Intell Eng Des Anal Manuf* 18(1):3–20
- Simpson TW, D'Souza BS (2004) Assessing variable levels of platform commonality within a product family using a multiobjective genetic algorithm. *Concur Eng* 12(2):119–129

- Simpson TW, Seepersad CC, Mistree F (2001) Balancing commonality and performance within concurrent design of multiple products in a product family. *Concur Eng* 9(3):175–190
- Simpson TW, Siddique Z, Jiao J (2006) Platform-based product family development. In: Simpson T, Siddique Z, Jiao J (eds) *Product platform and product family design*. Springer, New York, NY, pp 1–15
- Sodhi MS, Reimer B (2001) Models for recycling electronics end-of-life products. *OR Spektrum* 23(1):97–115
- SquareTrade (2008) SquareTrade Research: iPhone more reliable than BlackBerry, One Year In. SquareTrade. <http://www.squaretrade.com/pages/iphone-reliability-study-11-2008>. Accessed 15 Nov 2009
- SquareTrade (2009) SquareTrade Research: one-third of iPhones fail over 2 years, mostly from accidents. SquareTrade. <http://www.squaretrade.com/pages/iphone-reliability-study-06-2009>. Accessed 15 Nov 2009
- Thevenot HJ, Simpson T (2006) Commonality indices for assessing product families. In: Simpson T, Siddique Z, Jiao J (eds) *Product platform and product family design*. Springer, New York, NY, pp 107–129

# Chapter 29

## Application of the Generational Variety Index: A Retrospective Study of iPhone Evolution

Gopal Nadadur, Matthew B. Parkinson, and Timothy W. Simpson

**Abstract** The generational variety index (GVI) helps to identify the components of product variants that are most likely to require redesign in the future. These components can then be embedded with the flexibility required for them to be easily modifiable; the remaining components can be designed into a platform. This paper describes the application of the GVI technique in studying the evolution of the Apple iPhone, which was first released in 2007 and has since undergone multiple redesigns. The analysis includes the five generations of the iPhone (original, 3G, 3GS, 4, and 4S) and focuses primarily on mechanical subsystems. The results of the analysis and subsequent design recommendations are compared with the actual design evolution of the iPhone product line. For certain subsystems, this comparison reveals a divergence in Apple's design decision-making from the evolution recommended by the GVI technique. Limitations include its retrospective nature and the use of only publicly available data.

---

An earlier version of this chapter appeared in G. Nadadur, M. B. Parkinson, and T. W. Simpson (2012) Application of the Generational Variety Index: A Retrospective Study of iPhone Evolution, ASME Design Engineering Technical Conferences—Design Automation Conference, Chicago, IL, ASME, Paper No. DETC2012/DAC-70727 (© ASME 2012), reprinted with permission.

G. Nadadur

OPEN Design Lab, The Pennsylvania State University, University Park, PA 16802, USA

M.B. Parkinson

OPEN Design Lab Engineering Design, Mechanical Engineering, and Industrial Engineering, The Pennsylvania State University, University Park, PA 16802, USA

T.W. Simpson (✉)

Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA

Industrial and Manufacturing Engineering, Penn State University, University Park, PA 16802, USA

e-mail: [tws8@psu.edu](mailto:tws8@psu.edu)

## 29.1 Introduction

Certain product categories (e.g., consumer electronics, information technology, apparel) experience rapid changes in user needs and preferences, sometimes governed by rapid technological advancement. This can result in fast product turnover and short life cycles (Ko and Hu 2009). The technological and user-related factors force the company to evolve the designs of these products in the duration for which they are offered in the market (Sanderson and Uzumeri 1997). Increasing design and manufacturing efficiencies in these products requires the consideration of these factors during the design development process. The generational variety index (GVI) (Martin and Ishii 2002) was developed to address this specifically and to help companies to identify how their products may evolve as customer needs change. This paper applies GVI analysis in evaluating the multi-generational design of the Apple iPhone, thereby providing an interesting example of an evolving product which was designed without the conventional attention paid to consumer surveys, studies, assessments, and feedback (Isaacson 2011).

### 29.1.1 Evolving Designs

Product lines can undergo changes that are revolutionary or evolutionary (Axelsson 2009), both of which may be interrelated and interdependent. Revolutionary changes are effected relatively infrequently, when a new product line is being designed and launched. Since these design decisions have longer-lasting impact on the product line, they must involve the consideration of future progress of technology or temporal variations in user requirements. In contrast, evolutionary changes occur more frequently and result in incremental improvements in systems of functions within the product architecture. Despite the divergent semantics, a similar distinction is made by Haolun et al. (2009), which discusses product platform upgrades and evolution. Upgrades consist of incremental improvements in efficiencies or functionalities of the product. On the other hand, evolution, which results in more radical changes in the design, becomes necessary when instabilities (e.g., changing consumer needs/preferences and technology) cause disturbances in the steady-state condition of the market. A third study along these lines is Kivi et al. (2012), which categorizes improvements in product features as being either multi-generational or incremental.

A number of brands of products have fast turnover and short life cycles, but are offered in the market for a number of years. The consumer electronics segment is replete with examples of such products, which include smartphones (e.g., Apple iPhone, RIM Blackberry), cameras (e.g., Canon SLR), and calculators (e.g., Texas Instruments TI-83). In some cases (e.g., websites, pharmaceuticals), these products are strategically customized to suit *varying* technological and user requirements across the target markets (Wang and Zhou 2008). In other cases, in order to keep up

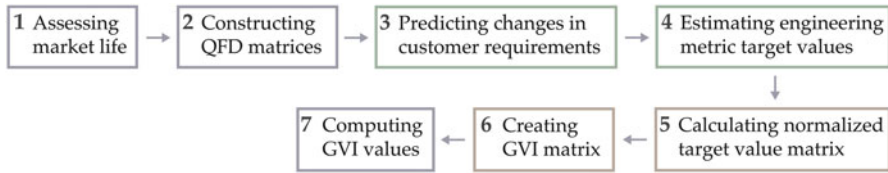
with *changing* technologies, user needs and preferences, production capacity, and market growth and demand, the designs and product lines of these brands are evolved in a variety of ways following their introduction (Haolun et al. 2009; Cao et al. 2010; Orbach and Fruchter 2011).

Numerous research efforts have sought to understand, quantify, and model the aforementioned factors and their impact on product evolution. For example, Cao et al. (2010) model the influence of changing demand and systems on the evolution of product function (i.e., the functions served by the product). Product evolution has also been modeled as a function of technological changes, market growth, and market preferences (Orbach and Fruchter 2011) and competitive market forces and technological changes (Zhang and Xu 2007), to mention two of the proposed modeling methods. Many of these models are intended for generating forecasts of product evolution for use in making present-day design decisions.

Product design entities tend to spend relatively low percentages of their research and development budgets on new product design and development (Rezayat 2000), which can be considered similar to revolutionary changes in product lines. A majority of the investment is on design reuse, either with parts that are completely unchanged or modified (i.e., evolved) in some manner. The process of evolutionary change in products can be made more efficient through the application of flexible design principles; the value of inbuilt flexibility is higher for greater levels of uncertainty in design decision-making (Nadadur et al. 2011).

Non-platformed products can be embedded with appropriate forms and amounts of flexibility to make their components easier to adapt and reuse across multiple generations of the designs (Van Wie et al. 2004) and to reduce the time and cost of designing existing products to keep pace with changing requirements (Keese et al. 2007). This goal can be achieved through multiple approaches, including modularity, parts reduction, spatial interface decoupling, and adjustability (Keese et al. 2007). In order to also increase production efficiency, the product and its assembly system can be concurrently designed through a method termed co-evolution (Bryan et al. 2010). Doing so allows the assembly system also to be adapted to evolving designs of the product; this decreases the time, cost, and effort involved in revamping the manufacturing system. Similarly, the recurring tasks across the different generations of the product can be forecast and integrated with stochastic product evolution models to identify ways in which to increase the life cycle of manufacturing systems (Ko and Hu 2009).

Flexibility can also be built into certain elements of product platforms (Ferguson et al. 2009) through techniques such as function-based product information modeling (Xu et al. 2006), multi-objective optimization based on predicted changes in functional and performance requirements (Lewis et al. 2011), change propagation analysis (Suh et al. 2007), and methods that are based on the generational variety index, which is described in the next subsection. Methods to assess this flexibility based on the design dependencies that arise from the physical interfaces in an architecture have been developed and are being studied elsewhere (Asikoglu and Simpson 2010). The value of such inbuilt flexibility increases with growth in levels of uncertainty in decision-making (Nadadur et al. 2011).



**Fig. 29.1** The seven steps in calculating generational variety index (GVI) values for the components or subsystems of a product platform (Martin and Ishii 2002)

### 29.1.2 Generational Variety Index

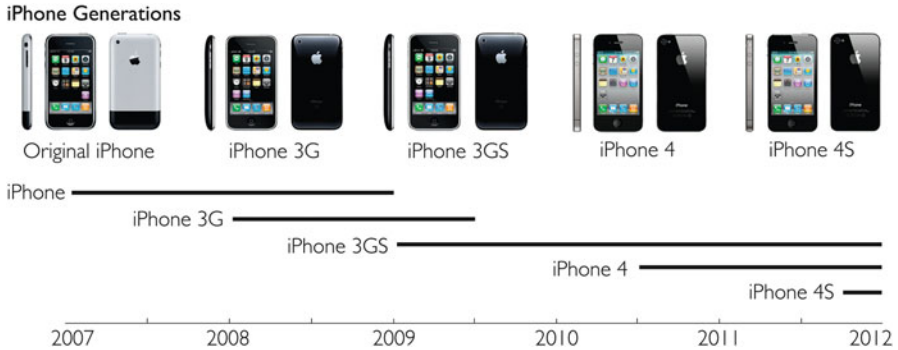
The generational variety index (GVI) (Martin and Ishii 2002) is a means to identify the components and subsystems within a product platform that will require significant modifications across multiple generations of the product. The technique is applied during the design planning phase of the platform.

As illustrated in Fig. 29.1, the technique involves seven basic steps (Martin and Ishii 2002). First, the designer estimates the market lifetime of the platform. Second, quality function deployment matrices are used to map generally stated customer requirements to more specific engineering metrics and then to the components necessary to achieve those specifications. Third, changes in customer requirements are forecast for the lifetime of the platform. The fourth and fifth steps entail the prediction of engineering metric specifications to be achieved by each generation of the platform and the subsequent normalization of these specifications to calculate the normalized target values of the metrics. Finally, in the seventh step, the GVI matrix is created and used to assess the GVI values for each subsystem of the platform. The GVI values thus calculated bear ordinal and ratio relationships to each other and are indicative of the percentage levels of change that their respective subsystems will require; the higher the GVI value, the greater the level of required change.

Since its introduction, the generational variety index has been incorporated into a number of methods and tools for product platform design. An example of these applications is the platform architecture utilized by a Dutch house-building company; the standardization of certain modules of the houses (e.g., structure, dimensions of floor slabs) allows the company to efficiently provide customers with the desired degree of variety (Veenstra et al. 2006). GVI is used in a conceptual design framework to leverage the benefits of product modularity, design for assembly, and design for variety (Gupta and Okudan 2008). The framework results in modules that are designed to require only a few of them to be replaced for the product to keep pace with changing user requirements. Additionally, the recommended designs are ranked in order of minimum assembly time. This framework is demonstrated through the design of an electronic toothbrush.

Simpson et al. (2011) presents a method that integrates product family design techniques that are qualitative (e.g., market segmentation grid, GVI) and





**Fig. 29.2** A timeline showing the periods during which each of the five generations of the iPhone has been available in the market

quantitative (e.g., multi-objective optimization, commonality indices). The method is demonstrated in the context of unmanned aerial vehicle design. An important observation is that the designs of GVI-based product families are close to the Pareto optimal designs. A similar conclusion is arrived at by Bobuk et al. (2010), which states that while GVI analysis does not necessarily result in an optimal product platform in terms of performance metrics, the analysis undoubtedly helps increase commonality across the variants of the evolving product.

### 29.1.3 Case Study Focus

The Apple iPhone is an example of a product that has been on the market for some time (since 2007), has attained global popularity, and has undergone both evolutionary and revolutionary changes since its release (see Fig. 29.2). The evolutionary changes include the transition from the original iPhone through the iPhone 3 and to the iPhone 3GS and subsequently from the iPhone 4 to the iPhone 4S. The main revolutionary change is the transition from the iPhone 3GS to the iPhone 4; examples of revolutionary alterations to the design include the improved display resolution, the transition to Apple-manufactured processors, the expansion to include CDMA technology, the introduction of a front camera, and the modified casing with a built-in antenna. More information about these transitions is presented in the following section.

Another important uniqueness in the development of the iPhone is Apple's apparent omission from the design process of public opinion measured through consumer surveys, feedback, focus groups, etc. (Isaacson 2011). Major decisions concerning the functionality and aesthetics of products were made by only a few individuals, based on their intuitive awareness of product characteristics that would be appreciated and valued by users. User feedback was limited to the appraisals of product prototypes by a few employees of the company before the final version was

released to the market. This process is contrary to most product design efforts, which entail a variety of formal methods of understanding the user populations' needs and wants. This paper presents a retrospective application of the generational variety index on publicly available information about five generations of the iPhone: original, 3G, 3GS, 4, and 4S.

## 29.2 Methodology and Case Study

This section describes the analysis of the five generations of the Apple iPhone using the GVI technique. There were a few constraints in this study. First, only publicly available information about the iPhone was utilized, and the analysis did not involve a detailed, component-level exploration of the product. Second, the five generations of the iPhone were examined retrospectively based on published details of its technological evolution; among other things, this involved seeking feedback from users who were already aware of the product and its different generations. Third, marketing research was not employed to understand, model, and predict consumer requirements. Fourth, the study was approached from a mechanical and industrial design perspective, resulting in relatively lower focus on concerns such as software and electronic components. Despite these constraints, the study is believed to be an interesting examination of the iPhone from a mechanical/industrial engineering standpoint.

### 29.2.1 Step 1: Assessing Market Life

As illustrated in Fig. 29.2, the original iPhone was released in early 2007. The dates of market entry of the subsequent generations of the product are early 2008 (iPhone 3G), early 2009 (iPhone 3GS), mid-2010 (iPhone 4), and late 2011 (iPhone 4S).

### 29.2.2 Step 2: Constructing QFD Matrices

The first stage of Step 2 was the listing of customer needs. This was done by soliciting the opinions of a group of students at the Pennsylvania State University. The question put before them was: "What characteristics or features do you desire in a smartphone?" The responses helped to develop the list of requirements shown in the first column of Fig. 29.3. The responses and engineering requirements are grouped into a few categories (e.g., display, sound, data download/transfer) to identify subsystems and thereby simplify the analysis.

The next stages of Step 2 involved first mapping the customer needs to engineering metrics (see Fig. 29.3), then mapping the engineering metrics to product subsystems. The 22 engineering metrics and 14 product subsystems are listed in Fig. 29.4.

	Display		Touchscreen		Sound		Processor and memory		OS Apps		Data download/transfer				Internet & connectivity		Capacity (power, battery, etc.)		Interface		Camera		Casing housing parts		Special features		
	size	resolution	technology	audio	mic	speaker	processing speed	memory capacity	OS	Apps	GSM & CDMA	processor	bandwidth	download/transfer speed	WiFi	Bluetooth	battery	quick charge	long usage time	charging interface	connector cable	resolution	video	housing	parts	special features	change
Display	X																									L	
Large size display		X																									M
High-res display			X																								L
Easy touchscreen operation				X																							M
Scratch-resistant screen					X																						L
Sound																											M
Good quality				X	X																						L
Support different music file formats									X																		M
Fast processing							X																				H
Large storage capacity								X																			H
Data transfer/download										X																	M
Fast									X																		M
Easy to sync											X																M
Connectivity											X																M
Multiple carriers										X																	M
Good coverage										X																	M
Easy WiFi connectivity									X																		M
Easy web browsing									X																		L
Easy emailing facility									X																		L
Easy video viewing		X							X																	M	
OS and applications									X																		H
Cool and versatile OS								X																			H
Wide selection of apps								X																			H
Easy app download and install								X																			H
Easy to customize and update								X																			L
Easy to customize and update								X																			L
Multitasking capability								X																			M
Power Management																											M
Simple charging																											L
Short charging time																											L
Long battery life																											L
Easy to sync																											L
Easy to sync																											L
Customizable regions		X	X																								L
Easy typing																											L
Voice activation																											L
Global positioning (GPS)																											L
Good front & back camera																											H
Overall sleekness (aesthetic)		X																									H
Reliable																											H
Robust and sturdy																											L
No software glitches									X																		L

Fig. 29.3 The QFD 1 matrix in Step 2 of the GVI process. The expected range of change of the customer requirements are classified as low (L), medium (M), and high (H)

Engineering Metric Target Matrix		Note: these changes have been assumed from a purely engineering design (and not software development) perspective																	
Display	no change	minor changes	Touchscreen	simple design changes	partial redesign	major redesign	Sound	Processor and memory	OS	Apps	Data download/transfer	Internet & connectivity	Capacity (user-expandable)	Interface	GPS	Cameras	Casing/housing parts	Charging interactive parts	
Current Market iPhone, A2007	3.5" glass TFT LCD, 32:10:6 aspect ratio @16.5kpps			capacitive multi-touch diagonal (BC9377) A	Wolfson Microelectronics W8105365		620MHz (reworked to 412MHz) Apple A712947, 128MB, 128MB, DMM1	4.5, 16 GB NAND flash	iOS 1.0	Net available	3298 NOK base CDMA 1xEV-DO Rev. A	802.11bg	3.7V / 1100mAh Li-Ion polymer not soldered to casing	30-pin dock connector + adaptor	No	Rear: 2MPix, video feed lens	Rear no video	13k aluminum, back/wide back/wide	volume: rectangular, elevated regions for mouse and trackball above volume buttons, home circular, bottom center, front panel headphone jack, bottom side antenna, internal camera circular, top right, back panel (rear)
Future Market iPhone 3G, 72008	3.5" glass TFT LCD, 32:10:6 aspect ratio @16.5kpps			capacitive multi-touch diagonal (BC9377) B	Wolfson Microelectronics W8105365	wider range of frequencies than iPhone	620MHz (reworked to Apple A712977), 128MB, 128MB, DMM1	8, 16 GB NAND flash	iOS 2.0 released 7/2008	100 apps	1298 NOK Tri-band 3.6Gbps UPTVS HSDPA	802.11bg	3.7V / 1200mAh Li-Ion polymer not soldered to casing	30-pin dock connector + adaptor	Yes	Rear: 2MPix, video feed lens	Rear no video	13k, glass, plastic, steel blank/white	headphone jack, flush-mounted
Future Market iPhone 3GS, 72009	3.5" glass TFT LCD, 32:10:6 aspect ratio @16.5kpps			capacitive multi-touch diagonal (BC9377) C	Crus Logic CYL6141	lower range of frequencies than iPhone 3G, wider than the iPhone	639MHz (reworked to Apple A712977), 256MB, 256MB, DMM1	8, 16, 32 GB NAND flash	iOS 3.0 released 6/2009	65,000 apps	1298 NOK Tri-band 7.2Mbps UPTVS mobile DSR HSDPA	802.11bg	2.1+H2DR Broadcom 4325	30-pin dock connector + adaptor	Yes	Rear: 3MPix, video auto focus	Rear: VCA @30frames/sec	13k, glass, plastic, steel blank/white	no change
Future Market iPhone 4, 82010	3.5" glass IPS LCD, 32:10:6 aspect ratio @32kpps			capacitive multi-touch diagonal (BC9377) D	Crus Logic CYL6141	same as iPhone 3GS	639MHz (reworked to Apple A712977), 512MB, 512MB, DMM1	8, 16, 32 GB NAND flash	iOS 4.0 released 6/2010	225,000 apps	GSX-12898 Qualcomm Snapdragon 12898 mobile DSR HSDPA, CDMA, EV-DO Rev. A	802.11bg	2.1+H2DR Broadcom 4325	30-pin dock connector + adaptor	Yes	Rear: 5MPix, video auto focus	Rear: 2MPix, VCA @30frames/sec	13k, glass, plastic, aluminum/steel blank/white	volume: two separate, circular buttons, multi more rectangular into metal enclosure, environmental camera, circular to be the car square (front)
Future Market iPhone 4S, 82011	3.5" glass IPS LCD, 32:10:6 aspect ratio @32kpps			capacitive multi-touch diagonal (BC9377) E	Crus Logic CYL6141	same as iPhone 3GS	Dual Core Apple A5, 512MB, 512MB, DMM1	16, 32, 64 GB NAND flash	iOS 5.0 released 10/2011	500,000 apps	GSX-12898 Qualcomm Snapdragon 12898 mobile DSR HSDPA, CDMA, EV-DO Rev. A	802.11bg	2.1+H2DR Broadcom 4325	30-pin dock connector + adaptor	Yes	Rear: 8MPix, video auto focus	Rear: 2MPix, VCA @30frames/sec	14k, glass, plastic, aluminum/steel blank/white	no change

Fig. 29.4 The engineering metric target values for the current and future target markets of the iPhone

### ***29.2.3 Step 3: Predicting Changes in Customer Needs***

In a forward-looking GVI analysis, this step would entail studying past and current trends in the market, developing predictive models of these trends, and utilizing these models to estimate future changes in customer needs. However, in this retrospective study, the actual rate of evolution of the technology and features were considered indicative of the rate of change of customer requirements. These rates were grouped into low, medium, and high categories and are shown for the different customer requirements in the last column of Fig. 29.3.

### ***29.2.4 Step 4: Estimating Engineering Metric Target Values***

The engineering metric target values used to track the evolution of the iPhone (see Fig. 29.4) were obtained from sources such as iSuppli (2012) and iDownload (2012). This information consisted of a mix of quantitative and qualitative technical specifications for every engineering requirement for each generation of the iPhone; every successive generation was considered a future market for the designer to take into account.

For instance, the engineering metric targets for the “operating system” engineering requirement were specified for the current market (original iPhone) and four future markets (iPhone 3G, 3GS, 4, and 4S); these targets were the iOS 1.0, 2.0, 3.0, 4.0, and 5.0, respectively. Another example is the “display resolution” engineering requirement, for which the engineering metric target was  $320 \times 480$  pixels at 163 pixels per inch until the iPhone 4, after which it was  $960 \times 480$  pixels at 326 pixels per inch. As stated earlier in this section, the level of detail of this information was limited by the amount of publicly available information about each engineering requirement.

While there may be some level of uncertainty associated with any predictions of the future technological changes, such predictions are not unrealistic. Intel is an example of a major organization that utilizes forecasts of technological advancements to guide the company roadmap (Intel 2012).

### ***29.2.5 Step 5: Calculating Normalized Target Value Matrix***

This calculation is an optional step in the GVI procedure (Martin and Ishii 2002) and was skipped in this study.

### ***29.2.6 Step 6: Creating GVI Matrix***

The matrix created at the end of Step 2 was used as the basis for the creation of the GVI matrix. This step involved assessing the complexity of subsystem-level redesign to satisfy the expected changes in the engineering requirements over the market lifetime of the product. A 0, 1, 3, 6, and 9 scale was used to indicate whether each of the relevant subsystems would require no redesign, few minor changes, numerous simple changes, partial redesign, or major redesign, respectively. The result of this process is shown in Fig. 29.5.

There can be two categories of redesign. The first category could refer to improvements in the existing functionalities of the product, e.g., the changes in the display and touch screen subsystems (Fig. 29.4). The second category involves changed or added functionalities of the design. The camera subsystem is an example of the second category, since its functionalities have grown from only a rear camera with no video capability to a front and rear camera combination with video capability. The GVI process is applicable to both types of redesign, as evidenced by this study.

### ***29.2.7 Step 7: Computing GVI Values***

The final step of the GVI procedure was the summation of the ratings assigned to each subsystem in the preceding step; this yielded a GVI value for every subsystem. These values are listed in the bottom row of Fig. 29.5.

## **29.3 Discussion**

This section discusses certain aspects of the different stages of the procedure applied in the previous section and examines the GVI values calculated for the iPhone subsystems and their implications on the design strategy of the product.

The construction of the list of customer requirements in Fig. 29.3 was based on feedback elicited from a group of students. Not all these students were smartphone users. However, even the non-smartphone users were aware to the functionalities of the iPhone and are likely to have been influenced by this knowledge while responding to the informal survey question. The iPhone is recognized as a product that has revolutionized consumers' perceptions of smartphone capabilities and user interaction. This is particularly true for the younger, college-age population of consumers. The work could be expanded upon by conducting the survey among more diverse populations (e.g., older people, people in lower income brackets); these responses are likely to result in a dramatically different list of consumer requirements.

	Display	Touchscreen	Sound	Processor	DRAM memory	Flash memory	Data transfer	Internet & connectivity	Software	Battery	GPS	Cameras	Outer casing	Physical interfaces
Display size	1	1											1	
Display resolution	3	3												
Touchscreen size	1	1											1	
Touchscreen technology		6												
Audio quality			3											
Processing speed				9	1									
Memory capacity					1									
Operating system				6	1	1		3						
Applications ("Apps")				3	1	1		3						
GSM/CDMA							9							
Frequencies							3							
Baseband processor							3							
Baseband memory							1							
Download/transfer speed							3	3						
WiFi speed								1						
Bluetooth								1						
Battery performance										1				
GPS											6			
Camera resolution									3					
Video capability	1								3			6		
Casing (housing parts)									3			1	6	3
Casing (interactive parts)												1	3	3
GVI	6	11	4	18	3	3	19	5	15	1	6	9	11	6
Scaled GVI (%)	32	58	21	95	16	16	100	26	79	5	32	47	58	32
#changes	1	2	4	3	2	2	4	1	4	4	1	3	2	2
Scaled #changes	25	50	100	75	50	50	100	25	100	100	25	75	50	50

**Fig. 29.5** GVI analysis of the iPhone. The engineering metrics and product subsystems are listed in the first column and first row, respectively. The calculated GVI values are compared with the number of times each subsystem has changed through the five generations of the iPhone; this count is based on Fig. 29.4. Also included are the scaled GVI and #changes; these are obtained by dividing each value by the maximum GVI or #changes, respectively

The mapping of the 22 engineering requirements to the 14 product subsystems (see Fig. 29.5) represents a higher-level analysis of the product than the standard engineering requirements to product components mapping procedure. This was due to a number of reasons, including the lack of detailed information about individual components; the sheer number of individual components due to the complexity of the product; and the desire to conduct a more holistic engineering design analysis of the product from a novel mechanical/industrial engineering perspective. The subsystems-level analysis is believed to be effective in achieving this end.

The aforementioned reasons are also why the ratings assigned in the GVI matrix (see Fig. 29.5) are relatively subjective. Information about costs, components, and underlying technology would have allowed for more objective assignment of ratings. This notwithstanding, the breadth of knowledge of the authors meant that multiple perspectives, viewpoints, and pieces of knowledge were accounted for in the GVI analysis. This is in keeping with the general procedure described by Martin and Ishii (2002).

Despite these limitations, the generational variety index evaluation of iPhone evolution has yielded some interesting observations. Three subsystems are identified as having higher GVI values ( $\geq 5$ ), while five subsystems have lower values ( $< 5$ ); these boundary values were chosen based on the extent of variation of the 14 GVI values. Higher GVIs imply a greater probability of redesign over the product's market lifetime. Accordingly, the "processor," "data transfer," and "software" subsystems are more likely to require future redesign. In contrast, the lower GVI subsystems—"sound," "DRAM memory," "flash memory," "Internet and connectivity," and "battery"—are more likely to remain unchanged throughout the iPhone's market lifetime based on the GVI analysis. These GVI predictions are consistent with the changes observed in the iPhone design over the years.

The scaled GVI and #changes metrics included in Fig. 29.5 allow for a high-level comparison of GVI predictions and actual changes in the design. The values of the two metrics are similar for certain subsystems: "display," "touch screen," "data transfer," "Internet and connectivity," "GPS," and "outer casing." For these subsystems, the calculated GVI value provides an accurate assessment of the future changes in the iPhone design. In contrast, marked divergences in the values of the two metrics are observed with some subsystems ("sound" and "battery"), indicating that the GVI assessment is not guaranteed to be accurate.

Since its introduction, the iPhone operating system (iOS), which is included in the "software" subsystem in this study, has been the platform for an ever-growing suite of Apple products. As explained by Nadadur et al. (2012), the iOS is a flexible platform, which is able to easily adapt to changing user-, business-, and regulation-related requirements. The lack of detailed knowledge makes it difficult to identify the application of a similar strategy with the other high-GVI subsystems, but it is likely that this is the case.

The iOS-based flexible platform strategy is a deviation from the guideline of platforming the low-GVI components or subsystems and modularizing the high-GVI components or subsystems. The success of this approach is evidenced by iOS's 52 % worldwide market share of the mobile and tablet operating system domain



(Netmarketshare 2012) and by the fact that the suite of Apple products served by iOS has grown from just the iPhone and iPod Touch to the iPhone, iPod Touch, iPad, and Apple TV (Nadadur et al. 2012). This is yet another example of the unconventional design approach that Apple is known for.

It should be noted, however, that such unconventional processes may not be universally applicable and could be associated with higher levels of risk. For instance, quickly evolving technology could result in increasing levels of difficulty in ensuring the compatibility of the different components of the product. GVI and other similar tools could prove useful in avoiding unexpected dysfunctionalities in the design due to mutually incompatible components.

## 29.4 Conclusions

There are a number of sources of limitations in this work. First, only publicly available data from websites and related sources were utilized. Second, the study was retrospective in nature and therefore applied already-known information about the evolution of the iPhone product line since its introduction in early 2007. Third, consumer research (e.g., marketing surveys, feedback datasets) were not drawn upon in the GVI analysis. And fourth, the analysis was approached from a mechanical/industrial engineering perspective, which implied a relative neglect of aspects such as the software.

The main utility of GVI is in providing designers with a quantifiable method to estimate the extent of future changes in the designs of products. This is useful in decision-making concerning the incorporation of flexibility into different components and subsystems. However, as discussed in the context of the iPhone study, GVI is not a foolproof method of predicting changes. A number of other methods have been proposed for the development of flexible products and platforms; some of these studies are mentioned in the Introduction section. A comparison of the effectiveness of these methods relative to GVI would make for interesting future research into the flexible designs.

**Acknowledgments** This study is not intended as an endorsement of Apple, Inc. or any of the company's products. The analysis reported in this paper utilized data that were publicly available thanks to the popularity of the iPhone product line. This research was partially funded by the National Science Foundation under Award No. 0846373. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

- Asikoglu O, Simpson TW (2010) A new approach for evaluating design dependencies in product architectures. In: 13th AIAA/ISSMO multidisciplinary analysis and optimization conference, No. AIAA-2010-9028
- Axelsson J (2009) Evolutionary architecting of embedded automotive product lines: an industrial case study. In: Software architecture and European conference on software architecture, IEEE/IFIP
- Bobuk A, Slingerland LA, Simpson TW, Donaldson B, Reichard K (2010) Validating the generational variety index (GVI) through product family optimization: a preliminary study. In: Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference. ASME International, No. DETC2010-285
- Bryan A, Ko J, Hu J, Koren Y (2010) Co-evolution of product families and assembly systems. CIRP Ann Manuf Technol 56(1):41–44
- Cao G, Guo H, Zhang C, Liu H, Li Q (2010) Function evolution and forecasting for product innovation. In: IEEE international conference on management of innovation and technology
- Ferguson S, Kasprzak E, Lewis K (2009) Designing a family of reconfigurable vehicles using multi-level multidisciplinary design optimization. Struct Multidiscip Optim 39(2):171–186
- Gupta S, Okudan GE (2008) Computer-aided generation of modularised conceptual designs with assembly and variety considerations. J Eng Des 19(6):533–551
- Haolun W, Liang H, Yuyi L (2009) Theoretical system of product platform innovation and evolution. In: 16th international conference on IEEE industrial engineering and engineering management
- iDownload (2012) The evolution of the iPhone. <http://www.idownloadblog.com/2011/08/23/iphone-evolution/>
- Intel (2012) Intel public roadmap for desktop, mobile, data center. <http://www.intel.com/content/www/us/en/processors/public-roadmap-article.html>
- Isaacson W (2011) Steve jobs. Simon & Schuster, New York, NY
- iSuppli (2012) iPhone 4S shows key design and component changes. <http://www.isuppli.com/Mobile-and-Wireless-Communications/News/Pages/iPhone-4S-Shows-Key-Design-and-Component-Changes.aspx>
- Keese DA, Tilstra AH, Seepersad CC, Wood KL (2007) Empirically-derived principles for designing products with flexibility for future evolution. In: Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference. ASME International, No. DETC2007-35695
- Kivi A, Smura T, Toyli J (2012) Technology product evolution and the diffusion of new product features. Technol Forecast Soc Change 79(1):107–126
- Ko J, Hu SJ (2009) Manufacturing system design considering stochastic product evolution and task recurrence. ASME J Manuf Sci Eng 131(5):051012
- Lewis PK, Murray VR, Mattson CA (2011) A design optimization strategy for creating devices that traverse the Pareto frontier over time. Struct Multidiscip Optim 43(2):191–204
- Martin MV, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. Res Eng Des 13(4):213–235
- Nadadur G, Garneau CJ, de Vries C, Parkinson MB (2011) A real options-based approach to designing for changing user populations of long-lifetime products. In: Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference. ASME International, No. DETC2011-48712
- Nadadur G, Kim W, Thomson AR, Parkinson MB, Simpson TW (2012) Strategic product design for multiple global markets. In: Proceedings of the ASME international design engineering technical conferences and computers and information in engineering conference. ASME International, No. DETC2012-70723
- Netmarketshare (2012) Mobile/tablet operating system market share. <http://netmarketshare.com/>

- Orbach Y, Fruchter GE (2011) Forecasting sales and product evolution: the case of the hybrid/electric car. *Technol Forecast Soc Change* 78(7):1210–1226
- Rezayat M (2000) Knowledge-based product development using XML and KCs. *Comput Aid Des* 32(5–6):299–309
- Sanderson SW, Uzumeri M (1997) *Managing product families*. Irwin, Chicago, IL
- Simpson TW, Bobuk A, Slingerland LA, Brennan S, Logan D, Reichard K (2011) From user requirements to commonality specifications: an integrated approach to product family design. *Res Eng Des*: 1–13. doi:[10.1007/s00163-011-0119-4](https://doi.org/10.1007/s00163-011-0119-4)
- Suh ES, de Weck OL, Chang D (2007) Flexible product platforms: framework and case study. *Res Eng Des* 18(2):67–89
- Van Wie M, Stone RB, McAdams DA (2004) Sustainable design through flexible product evolution. In: *Proceedings of the ASME international mechanical engineering congress and exposition*. ASME International, No. IMECE2004-60667
- Veenstra VS, Halman JJ, Voordijk JT (2006) A methodology for developing product platforms in the specific setting of the housebuilding industry. *Res Eng Des* 17(3):157–173
- Wang T, Zhou J (2008) Strategic choices of firms in expanding overseas business—a case study of Pfizer’s production scope evolution in China (1993–2002). *Front Bus Res China* 2(1):67–97
- Xu Q, Ong S, Nee A (2006) Function-based design synthesis approach to design reuse. *Res Eng Des* 17(1):27–44
- Zhang F-y, Xu Y-s (2007) Research on technical strategy for new product development based on TRIZ evolution theory. *Int J Prod Dev* 4(1–2):96–108

# Chapter 30

## Designing a Lawn and Landscape Blower Family Using Proactive Platform Design Approach

Keith Hirshburg and Zahed Siddique

**Abstract** In this chapter, a Scaling, Small Product, *Proactive Platform Design Method Using Modularity* (PPM) for Product Variations is used to create a product family of lawn and landscape blowers from the conception of an idea for the family, to the actual product variations.

### 30.1 Introduction

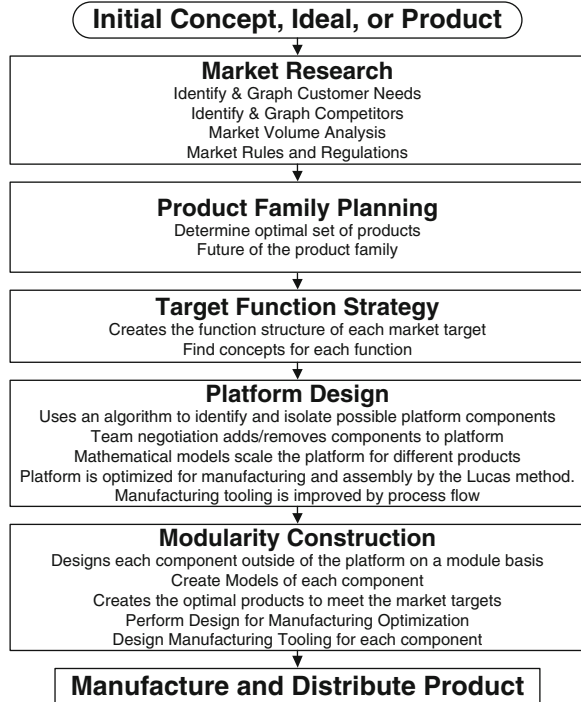
To be competitive in the current business environment, a company or engineering firm must be able to produce new products or designs in the market place with better quality and greater customization than competitors (Bower and Hout 1988; Stalk and Hout 1990). These business entities must also be able to accomplish this at a more strenuous pace than their competitors to capture the largest market share. Scaling, Small Product, *Proactive Platform Design Method Using Modularity* (PPM) for Product Variations (Fig. 30.1) uses the concept of modularity and scaling to assist the company to achieve higher competitive result. This chapter builds on already presented PPM in Chap. 8.

PPM begins with the initial ideal concept, followed by market research through direct and indirect customer interviews and also through competitor product analysis to identify the customer needs, to evaluate the importance in purchasing of each of the customer demands, to identify the competing products in the market segments and how well the competing products fulfill the customer requirements, and to determine what basic rules and regulations might dictate the product family. The next step in the method is *Product Family Planning*, where the product family is defined by product offering targets. These market targets include the optimal set

---

K. Hirshburg • Z. Siddique (✉)  
School of Aerospace and Mechanical Engineering, University of Oklahoma,  
Norman, OK 73019, USA  
e-mail: [zsiddique@ou.edu](mailto:zsiddique@ou.edu)

**Fig. 30.1** Case study road map



of products of the main market segments and market niches to create the greatest revenue from different customer basis. The step also involves planning for time to determine when the products and their updates will be delivered to the market, to properly fulfill the time between product family generations.

The next step in PPM is the *Target Function Strategy*, which maps out the function structure of each product offerings and then constructs theoretical concepts from individual functions. The next step, *Platform Design*, involves reviewing all possible function concepts and then uses an algorithm to identify and select a product family platform. Following the selection of the platform component, the actual design of the platform can be driven by using mathematical optimization to scale the component to meet the performance requirements of the different products. The platform is then improved for manufacturing, tools, and assembly using. Lastly, PPM uses one-to-one mapping of the non-platform components to integrate product variability through modularity.

A family of lawn and landscape blowers will be designed, based upon a scaled platform that is leveraged to support possible variants of hand blowers, backpack blowers, rolling blowers, fan assemblies, and beyond. The case study is used to demonstrate PPM, presented in Chap. 8, by creating a product line.

Since resources are not available to demonstrate the entirety of the method, market research phase will be performed with generated data for the customer needs, data on the competitors will be limited to the companies Stihl and Echo,

market volume analysis will be generated, and searching for the rules and regulations for the product will not be conducted. The entire product family planning phase will be included. Target function strategy phase will include function structures, but not concept generation for each function. Platform design and optimization phase will include an algorithm to identify and isolate platform components, and negotiation will not be covered since a single designer is creating the example (Fig. 30.1). Since the product family will not exist in physical form, the analysis for stress in the components, computations for fluid dynamics, and the design for manufacturing tooling are not performed.

## 30.2 Step 1: Market Research

The first step of market research is to identify the market segments for the product line and determine the common functions. After searching, it was identified that the main market segments to the blower family should be the handheld blower, handheld blower/vacuum, backpack blower, and backpack blower/vacuum (Fig. 30.2).

The product family will compete within these segments with at least have one product offered. Since the main concept of the blower family is a fan or pump device to move air, and the primary market is the lawn and landscape community, there are other close market segments to the blower family: push lawn vacuum, push lawn rolling blower, debris loader, and pulled trailer debris loader. These are close market segments, hence the team should attempt to create versions of the product family to offer in these segments as well, but they should not have the main emphasis. Design involving products for segments outside of the four main market segments are not included in the case study. Products outside the market segment are usually released after the initial product offerings. Using solely the concept of a fan or pump device to move air, other segments can be found: AC/heater fans, home office fans, hair dryers, car wash driers, car turbo, and compressors. These distant market segments can be viewed as possible to reach in the product family, but most likely reaching them will require the platform to produce lower performance. If the platform does not retain the proper performance in the four main segments, the distant segments should not be targeted. The market segment identification of the four main segments, the four close segments, and the distant segments are shown in Fig. 30.2 (the market segments have been created using ECHO, BillyGoat, and Contental).

After the market segments have been identified, the customer needs of each segment are determined through direct and indirect interviewing. The direct polling is conducted through physical interviews of the customer. The results for each feature are averaged and displayed in Fig. 30.3. The result shows customers are driven to the purchase of blowers by the features of air velocity, air volume, total weight, total fuel consumption time, safety, durability, gas powered, and being a backpack version. Figure 30.3 also shows that potential market niches may exist in using the features of low sound, high comfort, right-handed versions, annual












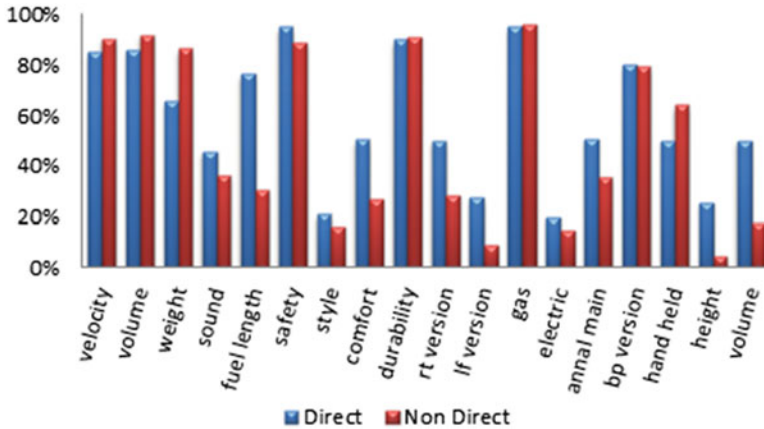
Market Segment Identification		
Main Segments	Close Segments	Distant Segments
<p>Hand Held Blowers</p> 	<p>Rolling Push Blower</p> 	<p>AC/Heater Fans</p> 
<p>Hand Held Blower Vacuum</p> 	<p>Rolling Push Vacuum</p> 	<p>Home/Office Fans</p> 
<p>Back Pack Blower</p> 	<p>Debris Loader</p> 	<p>Car Washer Driers</p> 
<p>Back Pack Blower Vacuum</p>  <p>Not actual vacuum.</p>	<p>Trailer Debris Loader</p> 	<p>Hair Drier</p> <p>Turbos</p> <p>Compressors</p>

Fig. 30.2 Market segment identification of the blower family

maintenance only, handheld, and small total volume. It can also be seen that very little emphasis should be placed on the features: high style and aesthetics, left-hand versions, electric versions, and small total height.

Along with the direct polling, indirect polling of the customers by e-mail was performed. The e-mail polling results produced extremes, which identify the most important features and the features that are the least important. The extremes with above 80 % were features: maximum air velocity, maximum air volume, minimum amount of weight, maximum amount of safety, maximum durability in the product, gas-powered version, and backpack version. The extremes with below 20 % were features: high style design, left-handed version, electric powered, minimum total height, and minimum total volume. The features outside of the two extremes were minimum sound, maximum fuel consumption allowance, comfort, right-handed version, and annual maintenance. The indirect polling results are shown in Fig. 30.3.



**Fig. 30.3** Combined direct and non-direct polling of the customer demands for the product family

Superimposing the direct and indirect polling indicates maximum air velocity, maximum air volume, total weight, safety of the device, durability of the device, gas powered, and being a backpack version, are the required customer demands when purchasing. Features that do not attract buyers are style of the design, left-hand version, electric-powered version, and total height of the device. Varying degrees of importance in the purchase of the product are the features: low sound output, maximum fuel consumption allowance, comfort, right-handed version, annual maintenance, handheld version, and total volume.

Market targets (Fig. 30.4) can be created using the demands found from the direct polling, non-direct polling, and the combined polling. A product should be offered to target each of the main market segments. Using the three most desired features from the interviews, the four market targets must have high standards of safety, high durability, and be gas powered. Fuel time, comfort, and volume are more of a concern for the backpack models in the design process since the product will be strapped to the users back.

The user will also dislike stopping to refill the gas tank, stopping to rest from poor comfort, or losing motor function from having a model with too large of volume. For the handheld models, total weight, a right-handed version, and the minimum total height are the important features. These features are important, because the total weight is held by hand instead of shoulders, the user is holding the product with their preferred hand, and total height is minimal so the product is not dragging across the ground. For the blower only units, the power of the blower is related to both the volume and velocity, creating the need for them to be optimized equally. In the blower/vacuum units, volume is the most important for performance, since the product will need to vacuum large bunches of leaves. The order of improvement, determined from surveys, is shown for different products in Table 30.1.



Market Targets		<ul style="list-style-type: none"> <li>• Very Safe</li> <li>• Good Durability</li> <li>• Gas Powered</li> </ul>	
		<b>Hand Held</b> <ul style="list-style-type: none"> <li>• Low Total Weight</li> <li>• Right Hand Version</li> <li>• Small Total Height</li> </ul>	<b>Back Pack</b> <ul style="list-style-type: none"> <li>• Larger Gas Capacity</li> <li>• Comfortable</li> <li>• Small Volume</li> </ul>
Blower	 <p>Optimize air velocity and air volume</p>	 <p>Optimize air velocity and air volume</p>	
	Vacuum	 <p>Optimize air volume</p>	 <p>Optimize air volume</p>

Fig. 30.4 Market targets with their respective features to optimize

Table 30.1 Order of features for market target

Order	Handheld		Backpack	
	Blower	Blower/vacuum	Blower	Blower/vacuum
1	Gas powered	Gas powered	Gas powered	Gas powered
2	Durability	Durability	Durability	Durability
3	Safety	Safety	Safety	Safety
4	Volume	Volume	Volume	Volume
5	Velocity	Total weight	Velocity	Fuel time
6	Total weight	Height	Fuel time	Comfort
7	Height	Velocity	Comfort	Velocity
8	Velocity	Sound	Sound	Sound
9	Fuel time	Fuel time	Volume	Volume
10	Comfort	Comfort	Height	Height
11	Annual main	Annual main	Annual main	Annual main
12	Style	Style	Style	Style

After the market targets have been decided, the competing products are evaluated. This evaluation is defined by the inclusion of the features, the quality of the features, how the product performs these features, and how many of these products are purchased. The values for each feature are calculated (Eq. 8.2) and is shown in Fig. 30.5. The feature values, closest to the customer demands, are what

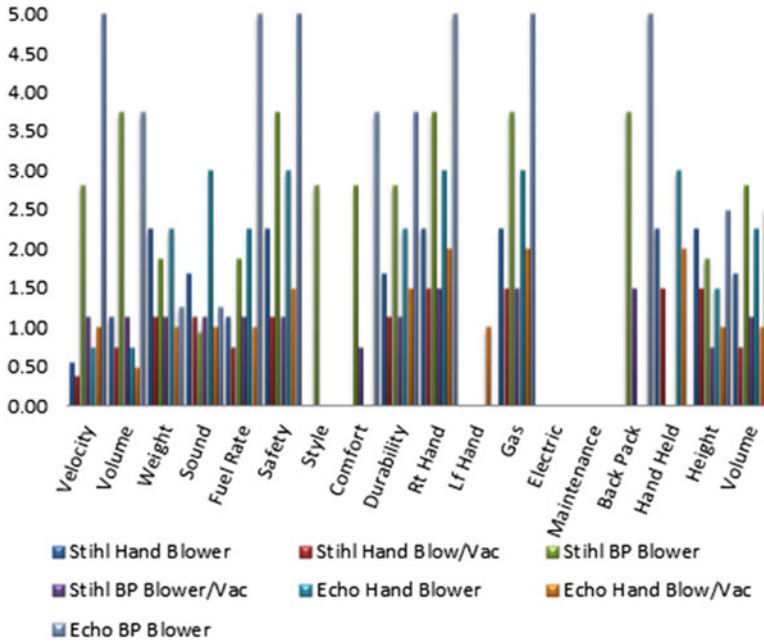


Fig. 30.5 The competitor product analysis of the market targets

the product family will need to outperform to sell the largest market share in the main segments. Reviewing Fig. 30.5, the models with best performance are the Echo hand blower, the Echo hand blower vacuum, the Echo backpack blower, and the Stihl backpack blower vacuum.

Fulfillment of customer needs in the market is evaluated by averaging together the competing products (Fig. 30.6). Features missing in the averaged scores are style, left-hand version, electric-powered version, and annual maintenance only. Comparison of the average fulfillment of customer demands by competitor’s products and the direct polling of customer requirements shows the demands in the market being met and the demands in the market not being met (Fig. 30.7).

Reviewing the comparison, features that are matched, meaning the customer demands have been met, are weight, sound, time of use by fuel tank, safety, style, gas powered, and handheld units. Since the demand for these features have been met by the current products in the market, the team will have to outdesign the competitors in these features to take market share away from them. The features offered in the competing products that have greater performance than the demand are right-hand version, total height, and total volume. These features will have less emphasis than the other features in the design process, since these demands are over met and outperforming them will not lead to greater sales. The features offered in the competing products that do not meet customer demand are output air velocity,

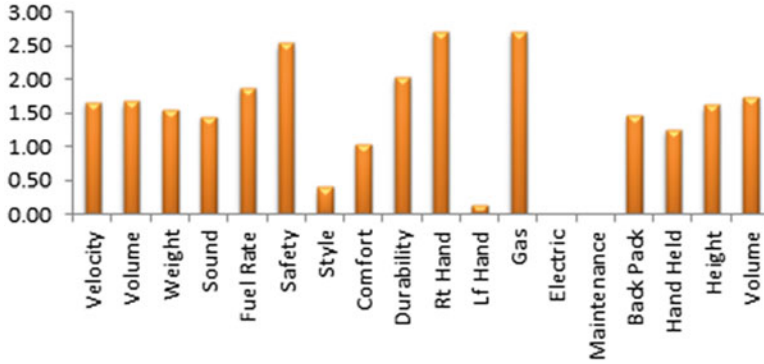


Fig. 30.6 Average fulfillment of customer demands in the market segment

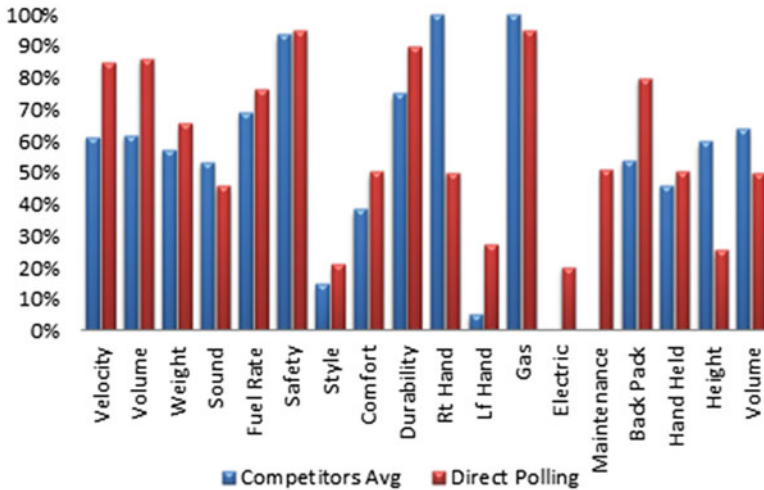


Fig. 30.7 Comparison of the competitors market targets versus the customer requirements

output air volume, durability, left-hand version, electric-powered version, and annual maintenance only version.

To obtain niche market share, the design of products in the product family should contain features of left-handed version, electric-powered version, and an annual maintenance only version. The features needed to obtain the largest amount of market share are output air velocity, output air volume, and durability. The product family should consist of right-handheld blower, right-handheld blower vacuum, right backpack blower, right backpack blower vacuum, left-hand backpack blower vacuum, and an electric-powered right-hand backpack blower vacuum.

### 30.3 Step 2: Product Family Planning

To plan the product family, the product targets need quantitative metrics to compare against the highest performance product family competitors (Table 30.2). To produce products that acquire market volumes in each segment, the products must outperform the competition on the features: output air velocity, output air volume, and durability. The products must also meet performance metrics in the other features.

To leverage the product family across the entire life span of the product generation, the product offerings must be planned and spaced out. The initial offering includes the four main market segment products: handheld blower, handheld blower/vacuum, and the backpack blower. After the initial product offerings, the plan should be to offer products at different time intervals until the next generation of product family is designed. In this product family, the life span was selected as 8 years due to the products life span of the competitors and the ability for the firm to design and release product generations. Using the 8-year life span of the generation, the market niches should be released to fulfill the middle time of the life span, and the product upgrades should be released in the second half of the life span. The product release times are shown in Fig. 30.8.

### 30.4 Step 3: Function Strategy

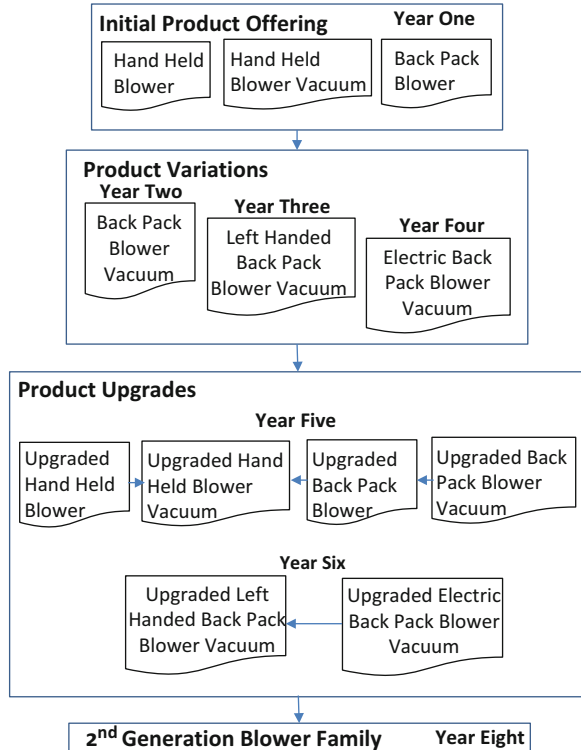
The function structure for the handheld blower uses four systems to create the functions of the product. The “Air to/from the Engine System” controls air going into the carburetor and the air leaving the engine. The “Engine System” controls the engine by metering the fuel into the engine cylinders, turning the drive shaft by the cylinder, and turning the turbine from the drive shaft. The “Engine Control System,” controlled by the user, is responsible for storing the gasoline for the engine, controlling the speed of the engine, turning on and off the engine, and starting the engine. The “Main Blower Process System” funnels air into the turbine housing, sucks air into the turbine, pushes air out of the turbine assembly, pushes air down the outlet tube, and pushes air out of the outlet tube. The handheld blowers’ function structure is shown in Fig. 30.9.

The function structure for the handheld blower vacuum uses five systems to create the functions of the product: Air to/from the Engine System, Engine System, Engine Control System, Main Blower Process System, and Main Vacuum Process System (Fig. 30.10). The “Main Vacuum Process System” is essentially the “Main Blower Process System” working in reverse. The handheld blower vacuums’ function structure is shown in Fig. 30.10.

Similarly, the function structure for the backpack blower uses four systems to create the functions of the product. The “Air to/from the Engine System” controls air going into the carburetor and the air leaving the engine. The “Engine System”



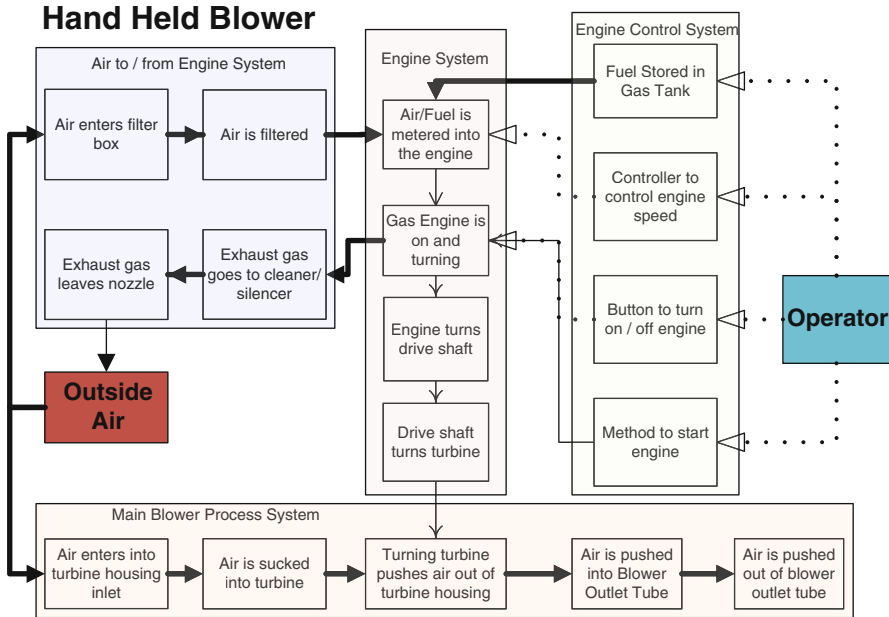
**Fig. 30.8** Product generation planning map



controls the engine by metering the fuel into the engine cylinders, turning the drive shaft by the cylinder, and turning the turbine from the drive shaft. The “Engine Control System,” controlled by the user, is responsible for storing the gasoline for the engine, controlling the speed of the engine, turning on and off the engine, and starting the engine. The “Main Blower Process System” funnels air into the turbine housing, sucks air into the turbine, pushes air out of the turbine assembly, pushes air down the outlet tube, and pushes air out of the outlet tube. The function structure for the backpack blower vacuum (right- and left-hand) uses five systems to create the functions of the product: Air to/from the Engine System, Engine System, Engine Control System, Main Blower Process System, and Main Vacuum Process System.

### 30.5 Step 4: Platform Design

The elements of the function structure are used to create a table with the component name, the product it is used in, the function, if it is scalable, if it is a standard or off-the-shelf part, and if the component can be modified. When the functions from the function structure are added to the sheet, a component is named for the function,



Component Name	Market Target	Component Function	Scalable	Standard Part / Off Shelf	Modified / Changed
air filter box	H H B	Air enters filter box	yes	no	yes
air filter	H H B	air is filtered	no	yes	no
muffler	H H B	exhaust gas goes to cleaner/silencer	yes	no	yes
exhaust piping	H H B	exhaust gas leaves nozzle	yes	no	yes
carburetor	H H B	air/fuel is metered into the engine	no	no	yes
gas engine	H H B	gas engine is on and turning	no	yes	no
drive shaft	H H B	drive shaft turns turbine (trans)	no	no	yes
gas tank	H H B	fuel stored in gas tank	yes	no	yes
cable throttle	H H B	controller to control engine speed	no	no	yes
switch	H H B	button to turn on/off engine	no	yes	no
pull cord	H H B	method to start engine	no	no	yes
housing inlet	H H B	air enters into turn housing inlet	yes	no	yes
turbine	H H B	air is sucked into turbine	yes	no	yes
turbine housing	H H B	turning turbine pushes air out of turbine housing	yes	no	yes
blower tube	H H B	air is pushed into blower outlet tube	yes	no	yes
tube nozzle	H H B	air is pushed out of blower outlet tube	yes	no	yes
handle	H H B	handle to hold unit	no	no	yes

Fig. 30.9 Function structure of the handheld blower market target

the function is marked if it can be scalable, the component is marked if it is an off-the-shelf part, and the function dimensions are decided if it can be modified. The handheld blower's input sheet (Fig. 30.9) contained 17 functions; the handheld blower vacuum's contained 20 functions (Fig. 30.10); the backpack blower's input sheet contained 18 functions; and the backpack blower vacuum's input sheet contained 21 functions. The algorithm, presented in Sect. 8.3.4, is used to identify

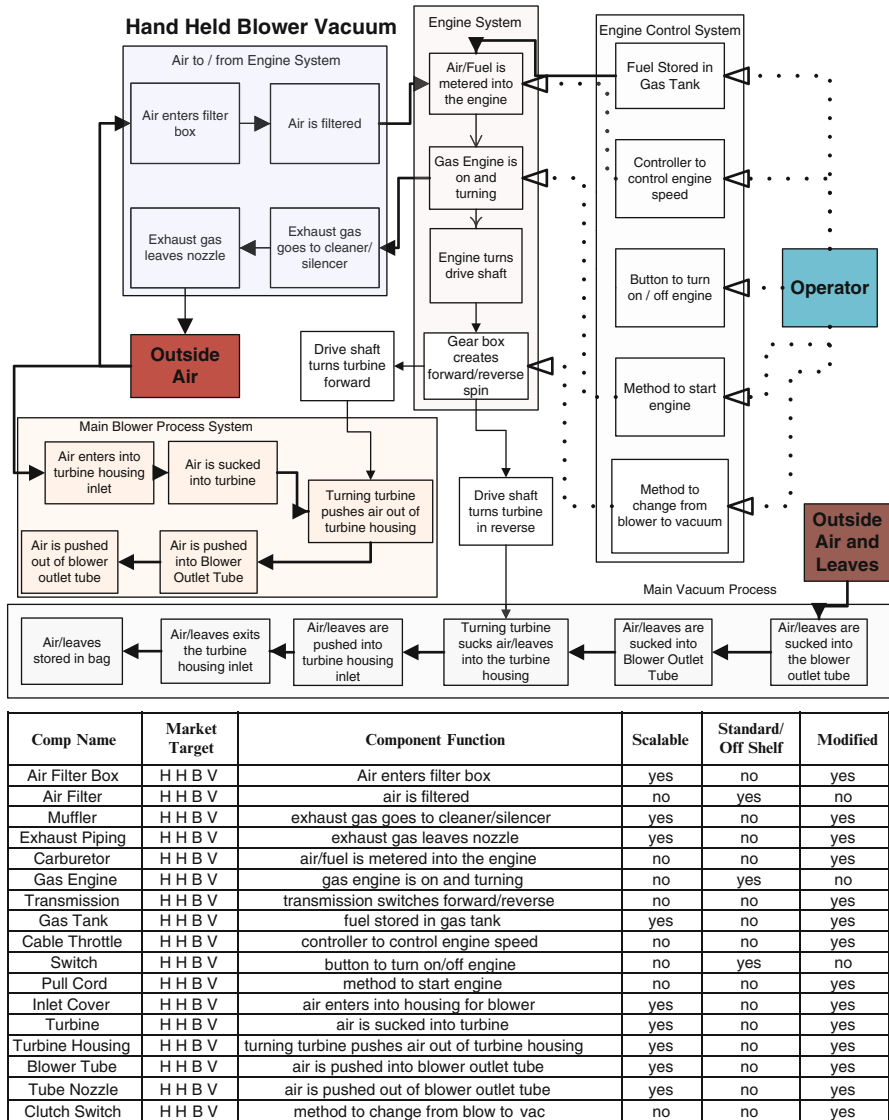


Fig. 30.10 Function structure for the handheld blower vacuum market target

and isolate a potential platform. Application of step one of the algorithm did not identify any component for the platform, this is due to no functions in the spreadsheets being described as: non-modifiable, non-off-the-shelf parts, and were present in at least 75 % of the products. Step two of the algorithm identifies the off-the-shelf components in at least 75 % of the products. Step two resulted in two functions being placed in the possible platform components list shown in



**Table 30.3** The possible platform components list for step two of the platform finding algorithm (components that are off-the-shelf parts or standard parts which are found in 75 % of the market targets and isolates them)

Component name	Market target	Component function	Scalable	Std. part/ off shelf	Modified/ changed
Air filter	4/4	Air is filtered	No	Yes	No
Switch	4/4	Button to turn on/off engine	No	Yes	No

**Table 30.4** Results of Step 3 of the platform component finding algorithm (components that share a function with at least 75 % of the market targets)

Component name	Market target	Component function	Scalable	Std. part/ off shelf	Modified/ changed
Air filter box	4/4	Air enters filter box	Yes	No	Yes
Muffler	4/4	Exhaust gas goes to cleaner/ silencer	Yes	No	Yes
Exhaust piping	4/4	Exhaust gas leaves nozzle	Yes	No	Yes
Gas tank	4/4	Fuel stored in gas tank	Yes	No	Yes
Cable throttle	4/4	Controller to control engine speed	No	No	Yes
Pull cord	4/4	Method to start engine	No	No	Yes
Turbine	4/4	Air is sucked into turbine	Yes	No	Yes
Turbine housing	4/4	Turning turbine pushes air out of turbine housing	Yes	No	Yes
Blower tube	4/4	Air is pushed into blower outlet tube	Yes	No	Yes
Tube nozzle	4/4	Air is pushed out of blower outlet tube	Yes	No	Yes
Air filter	4/4	Air is filtered	No	Yes	No
Switch	4/4	Button to turn on/off engine	No	Yes	No

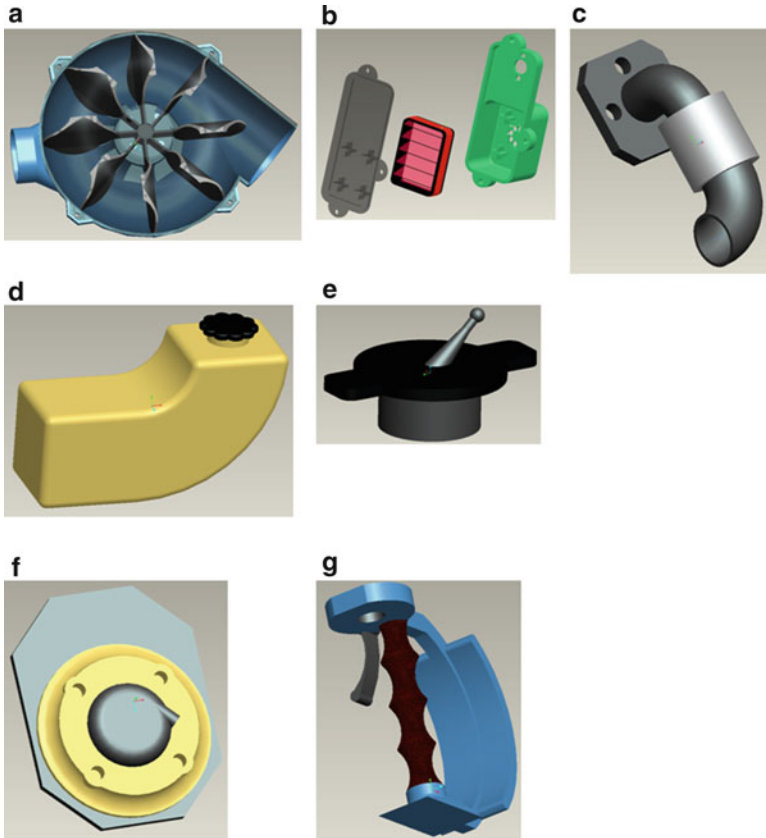
Table 30.3. Step three of the algorithm determines the components of the products that have a shared function in at least 75 % of the products (Table 30.4).

The last step in the algorithm determines the components of the platform that are scalable, the percentage of use in the product family by the components of the platform, and the components that need to be approved by the negotiation model (Table 30.5). The following components were identified as scalable platform components: air filter box, muffler, exhaust piping, gas tank, turbine, turbine housing, blower tube, and tube nozzle. In addition, non-scalable platform components have been identified as: cable throttle, pull cord, air filter, and on/off switch. The non-platform components are drive shaft, housing inlet, handle, inlet cover, transmission, back support, shoulder straps, clutch switch, storage bag, bag clamp, gas engine, and the carburetor.

**Table 30.5** Final results of the platform component finding algorithm

Component name	Market target	Component function	Scalable	Std. part/ off shelf	Modified
Scalable, 100 % platform components					
Air filter box	4/4	Air enters filter box	Yes	No	Yes
Muffler	4/4	Exhaust gas goes to cleaner/ silencer	Yes	No	Yes
Exhaust piping	4/4	Exhaust gas leaves nozzle	Yes	No	Yes
Gas tank	4/4	Fuel stored in gas tank	Yes	No	Yes
Turbine	4/4	Air is sucked into turbine	Yes	No	Yes
Turbine housing	4/4	turning turbine pushes air out of turbine housing	Yes	No	Yes
Blower tube	4/4	Air is pushed into blower outlet tube	Yes	No	Yes
Tube nozzle	4/4	Air is pushed out of blower outlet tube	Yes	No	Yes
Non scalable, 100 % platform components					
Cable throttle	4/4	Controller to control engine speed	No	No	Yes
Pull cord	4/4	Method to start engine	No	No	Yes
Air filter	4/4	Air is filtered	No	Yes	No
Switch	4/4	Button to turn on/off engine	No	Yes	No
Scalable, possible platform components (none)					
Non scalable, possible platform components (none)					
Non platform components					
Drive shaft	2/4	Drive shaft turns turbine (trans)	No	No	Yes
Housing inlet	2/4	Air enters into turn housing inlet	Yes	No	Yes
Handle	2/4	Handle to hold unit	No	No	Yes
Inlet cover	2/4	Air enters into housing for blower	Yes	No	Yes
Transmission	2/4	Transmission switches forward/ reverse	No	No	Yes
Back support	2/4	Pad that supports the unit of back	Yes	No	Yes
Shoulder straps	2/4	Strap that holds the unit onto the back	No	Yes	Yes
Clutch switch	2/4	Method to change from blow to vacuum	No	No	Yes
Storage bag	2/4	Air/leaves stored in bag	Yes	No	Yes
Bag clamp	2/4	Clamps bag onto housing inlet	Yes	No	Yes
Gas engine	1/4	Gas engine is on and turning	No	No	No
Carburetor	1/4	Air/fuel is metered into the engine	No	No	Yes

The algorithm identified only permanent platform components, so there is no need for negotiation to change the platform components from the algorithm results. The negotiation model also needs multiple designers to implement, and this case study only uses a single designer. The modeling and design of the components starts



**Fig. 30.11** CAD Model of components. (a) Turbine platform components, (b) air input system, (c) air exhaust system, (d) gas tank system, (e) engine on/off control system, (f) engine starter cord system, (g) engine throttle control system

with the most crucial platform components and works to the less crucial components.

According to market research, the velocity and volume of air is the most important feature, which is generated in the turbine assembly and all other components in the product either redirect or control the air. These components outside of the turbine assembly generate frictional losses creating a reduced velocity and volume of the air. Since the consumer demands are the full throttle performance, the flow of the air inside the blower can be considered as a one-dimensional steady-state flow. The turbine and turbine assembly are modeled to allow scaling of the diameter to create different air velocity and air volume for the different product variants. The turbine assembly is modeled in CAD (Fig. 30.11a) and can be scaled to the optimal values for each product.

The other scaling components in the platform are part of the air intake system, and the exhaust system, to allow different amounts of air into the engine depending on the engines' needs. The air filter assembly scales vertically and horizontally to allow for different size air filters. The air intake system (Fig. 30.11b), made up of an air filter, filter box, and filter cover, is modeled in CAD to allow for flexible on the fly changes. The air intake system also uses a modular connection to the carburetors by using two  $\frac{1}{4}$  in. bolts in  $180^\circ$  increments. The diameter of the exhaust tubing scales larger to allow for greater exhaust flow or scales smaller to keep exhaust back pressure to meet the needs of the different engines. The exhaust system (Fig. 30.11c) uses a modular connection with the engine by using three  $\frac{1}{4}$  in. bolts in  $90^\circ$  increments. The gas tank (Fig. 30.11d) scales in horizontal and vertical dimensions to match the scaling of the turbine assembly to offer a larger/smaller capacity of gasoline for the larger/smaller engines when the turbine diameter is larger/smaller.

The non-scaling platform components include engine starter system, on/off switch, and the engine throttle control. The non-scaled platform components are designed to fulfill their roles in the function structures. The engine starter system starts the engine by rotating the piston in the cylinder. The engine starter system (Fig. 30.11f) is connected onto the engine by 4  $\frac{1}{4}$  in. bolts in  $90^\circ$  increments. The on/off switch is an off-the-shelf part. The switch closes, or opens, the electric circuit powering the spark plug. The switch is shown in Fig. 30.11e. The engine throttle control system (Fig. 30.11g) is made up of a lever that the user moves to control the engine speed. This lever pulls a cable that is connected to the carburetor, and the carburetor meters the air fuel mixture into the engine. The lever is mounted on a handle that is used to hold up the handheld product versions, and the handle is used to direct the air output tube for the backpack product versions.

At this point in the design process of the platform, FEA is used to improve and optimize for stresses in the component during use, and CFD to validate the fluid flow. This case study does not perform this analysis. After the optimization of the designs for stress and fluid flow, the design is improved for manufacturing and assembly, through the use of the Lucas method (Sect. 8.3.5). The first step is to perform the functional analysis to determine the design efficiency [Eq. (8.3)], which is performed by dividing the components into essential (15) and nonessential components (15), which gives design efficiency of 50 % (Table 30.6). The design efficiency of 50 % is lower than the desired 60 %, but still reasonable. The functional analysis efficiency result can be improved by combining components or by using a smaller number of fasteners.

The feeding analysis is used to examine the handling of the components when assembling them to the platform. The Lucas method provides charts for the handling scores to be computed based on the size and weight of the component, the handling difficulties of the component, the end-to-end symmetry of the component, and the rotational symmetry of the component. The data for the feeding analysis of platform components is shown in Table 30.6. The calculated feeding ratio for the product platform is 2.35 and the desired feeding ratio is under 2.5.

**Table 30.6** Feeding and fitting analysis data for the platform components

Component name	Feeding analysis				Fitting analysis data				Access/ vision	Alignment	Force	
	Essential	Non-essential	Size/ weight	Handling difficulties	Orientation of part	Rotational orientation	Placing/ fastening	Direction				Insertion
Turbine	1	0	1	0	0	0	2	0	0	0	0	0
Turbine house top	1	0	1	0	0.1	0.2	1	0	0	0	0	0
Turbine house bottom	1	0	1	0	0.1	0.2	1	0	1.2	0	0	0
Turbine house fasteners	0	4	4	0	0.4	0	16	0	0	0	0	0
Air filter bottom	1	0	1	0	0.1	0.2	2	0.1	0	0	0	0.7
Air filter top	0	1	1	0	0.1	0.2	2.3	0.1	0	0	0	0
Air filter fasteners	0	2	2	0	0.2	0	8	0.4	0	0	0	0
Air filter	1	0	1	0	0	0	1	0.1	0	0	0	0
Exhaust	1	0	1	0	0.1	0.2	2	0.1	0	0	0	0.7
Exhaust muffler	1	0	1	0	0.1	0.2	2.3	0.1	0	0	0	0.6
Gas tank	1	0	1	0	0.1	0.2	2	0	0	0	0	0.7
Gas tank lid	1	0	1	0	0	0	4	0.1	0	0	0	0
Gas tank fasteners	0	4	4	0	0.4	0	16	0	0	0	0	0
Starter handle	1	0	1	0	0.1	0.2	2	0	0	0	0	0
Starter assy top	1	0	1	0	0.1	0.2	2	0	0	0	0	0
Starter assy bottom	1	0	1	0	0.1	0	1	0	0	0	0	0.7
Starter cord	1	0	1	0.6	0	0	2	0	0	0	0	0.6
Starter assy fasteners	0	4	4	0	0.4	0	16	0	0	0	0	0
Throttle handle	1	0	1	0	0.1	0.2	2	0	0	0	0	0
Throttle lever	1	0	1	0	0	0.2	2.3	0	0	0	0	0.6
Sums	15	15	30	0.6	2.5	2.2	86.9	1	1.2	0	2.8	1.8
Total index	Feeding ratio				Fitting ratio				89.1	5.94		
					35.3	2.35						

The fitting analysis examines the ability for the components to be inserted into the platform. The fitting analysis charts, Lucas Method, examine the part placing and fastening, process direction, insertion, access and vision, alignment, and force. The data for the fitting analysis of the platform components is shown in Table 30.6.

The fitting ratio of the platform products was calculated at a ratio of 5.94 which is significantly over the 2.5 goal ratio. With a fitting ratio of more than double the goal, the platform will undergo significant redesigns to improve for assembly. The platform is redesigned using a smaller number of fasteners, a greater number of platform components will be assemble with snaps, and each component will self-hold on to the assembly during insertion.

The last step in the Lucas method is the manufacturing analysis to evaluate complexity of the components. The manufacturing analysis' cost index is calculated by Eq. (8.6). The data for the manufacturing analysis of the platform components are shown in Table 30.7.

The manufacturing analysis is used to determine which components should be redesigned to lower the complexity. The components that have unusually high index, as in the turbine and exhaust, should be redesigned to lower the production cost. The manufacturing tooling design for the platform should also be optimized to allow efficient manufacturing of the platform components. Due to the product family not being created, the manufacturing tooling optimization is not conducted, but information of optimizing the manufacturing tool is explained in Sect. 8.3.4. The nonphysical platform for the product family is the design of the modular connections in the products. The connections use one to four  $\frac{1}{4}$  in. hex bolts in  $90^\circ$  increments. The nonphysical platform assists in the design of the connections for components and allows for more component variation to be interchanged in the products.

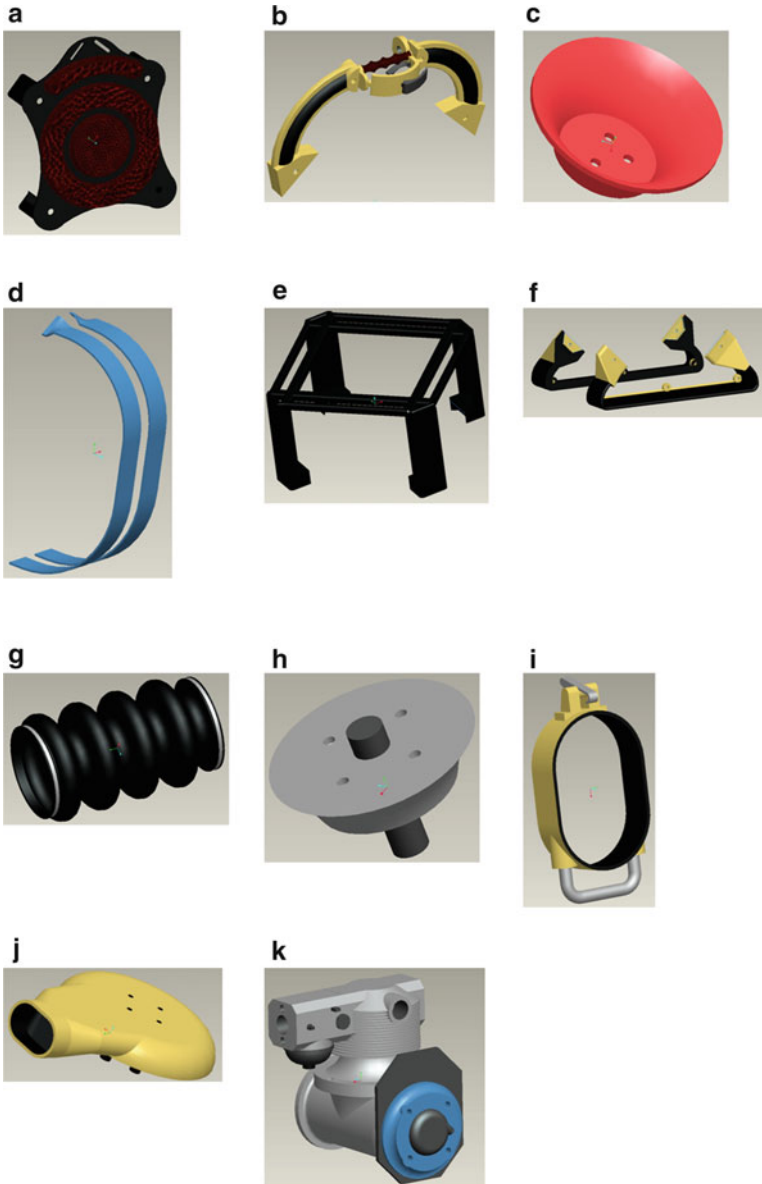
## 30.6 Step 5: Modularity Construction

The platform components need to be designed, optimized, and be able to perform to the four market targets (Sect. 30.2) by scaling. To foster modularity in the non-platform components, all of components are designed to be removed and interchanged with another component, and still have the product work correctly. Using a modular architecture, all of the non-platform components must connect to the product in a one to one manner. In addition, all component connections are made by one to four  $\frac{1}{4}$  in. bolts set at  $90^\circ$  increments.

The design requirements of the back pad for the backpack version are to mate to the turbine assembly, to support and comfort the users back, to be a connection point for the straps, and to create a sturdy structure for the product. The back pad will be included in the backpack blower, backpack blower vacuum, left-handed backpack blower vacuum, and the electric-powered backpack blower vacuum. Using these constraints, the back pads were designed and are shown in Fig. 30.12a. The design requirements for the handle, for the handheld versions,

**Table 30.7** Manufacturing analysis

Component name	Complexity	Material factor	Minimum	Processing	Tolerance/finish	Volume	Material cost	Waste	Index
Turbine	6.5	1.2	1	6.2	1.2	1	0.00341	1.4	58.04
Turbine housing top	1.2	1	1	11	1	1	0.00107	1.1	13.20
Turbine housing bottom	1.2	1	1	11	1	1	0.00107	1.1	13.20
Turbine housing fasteners	2	1.2	1	2.47	1.1	1	0.00068	2	6.52
Air filter bottom	1.2	1	1	11	1	1	0.00107	1.1	13.20
Air filter top	1.2	1	1	11	1	1	0.00107	1.1	13.20
Air filter fasteners	2	1.2	1	2.47	1.1	1	0.00068	2	6.52
Air filter	3	1	1	11	1	1	0.00058	1.1	33.00
Exhaust	6.1	1.5	1	5.7	1.2	1	0.00068	1.4	62.59
Exhaust muffler	2.5	1.5	1	5.7	1.2	1	0.00068	1.4	25.65
Gas tank	1.2	1	1	11	1	1	0.00107	1.1	13.20
Gas tank lid	1.2	1	1	11	1	1	0.00107	1.1	13.20
Gas tank fasteners	2	1.2	1	2.47	1.1	1	0.00068	2	6.52
Starter handle	1.2	1	1	11	1	1	0.00107	1.1	13.20
Starter assembly top	1.2	1	1	11	1	1	0.00107	1.1	13.20
Starter assembly bottom	1.2	1	1	11	1	1	0.00107	1.1	13.20
Starter cord	0	1	1	5	1	1	0.00035	1	0.0004
Starter assembly fasteners	2	1.2	1	2.47	1.1	1	0.00068	2	6.52
Throttle handle	1.2	1	1	11	1	1	0.00107	1.1	13.20
Throttle lever	1.2	1	1	6.2	1	1	0.00107	1.1	7.44

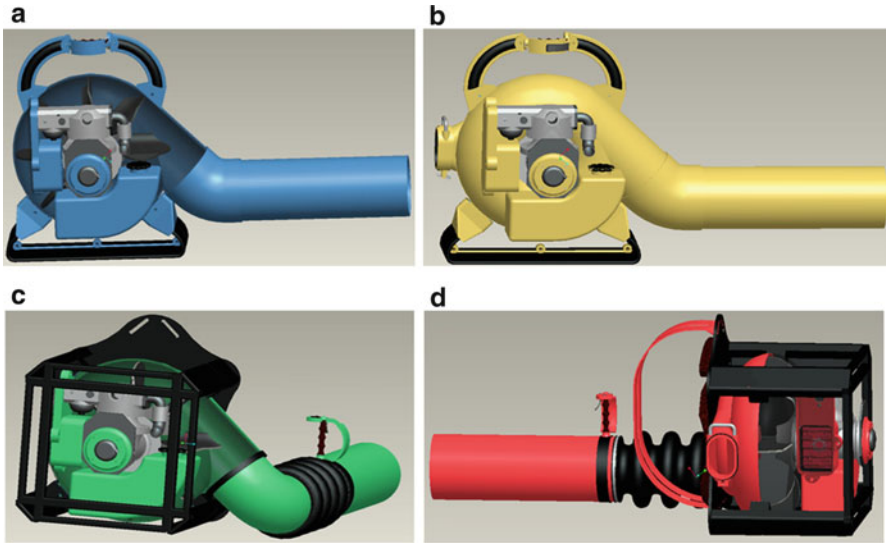


**Fig. 30.12** CAD models of variety modules. (a) Backpack pad, (b) handheld versions handle, (c) blower version's inlet cover, (d) shoulder strap used in the backpack products, (e) bash cage for backpack models, (f) bottom guard for the handheld versions, (g) bendable tubing for backpack versions, (h) drive shaft for the blower only models, (i) storage bag lock for the vacuum models, (j) vacuum outlet for the vacuum models, (k) small gas engine used in the models



are to mate to the turbine assembly, house the on/off switch, house the throttle system, be right handed, and hold the entire weight of the product. The handle will be used in the handheld blower and the handheld blower vacuum model. Using these constraints, the handle was designed and is shown in Fig. 30.12b. The design of the inlet cover for the blower only version requirements is to keep the user from getting hurt and to allow air into the turbine housing inlet. The inlet cover will be used in the handheld blower and in the backpack blower models. Using these constraints, the inlet cover was designed and is shown in Fig. 30.12c. The design of the shoulder straps for the backpack version requirements is to support the weight of the product on the users back comfortably. The shoulder straps (Fig. 30.12d) were designed for use with the backpack blower, left-handed backpack blower vacuum, electric-powered backpack blower vacuum and the backpack blower vacuum models. The design of the bash cage for the backpack versions' requirements is to protect the unit from being damaged in a drop and to keep the unit straight up when set down. The bash cage (Fig. 30.12e) will be used in the backpack blower and in the backpack blower vacuum models. The design of the bottom guard for the handheld only version requirements is to protect the unit from being damaged in a drop, to keep the unit straight up when set down, and to keep the total height of the unit below 18 in. The bottom guard will be used in the handheld blower and in the handheld blower vacuum models (Fig. 30.12f). The design of the bendable tubing for the backpack version requirements is to allow for the user to manage the direction of the blower tube. The bendable tubing will be used in the backpack blower and in the backpack blower vacuum models (Fig. 30.12g). The design of the drive shaft for the blower only version requirements is to keep the turning of the engine mated with the turbine and also to be housed in the same place as the transmission. The drive shaft will be used in the handheld blower and in the backpack blower models (Fig. 30.12h). The design of the storage bag lock for the vacuum version requirements is to keep the user from getting hurt and to allow quick change to storing the vacuumed material. The storage bag lock will be used in the handheld blower vacuum and in the backpack blower vacuum models (Fig. 30.12i). The design of the vacuum outlet for the vacuum version requirements is to keep the user from getting hurt and to allow air to leave the turbine housing and be collected in a bag. The vacuum outlet will be used in the handheld blower vacuum and in the backpack blower vacuum models (Fig. 30.12j). The design of the engine for the units is to keep provide the unit with power to meet its performance and to fit in the required space allocation. The engine will be used in all models (Fig. 30.12k).

After the design and modeling of the components, the CAD model for the varieties were assembled: handheld blower (Fig. 30.13a), handheld blower vacuum (Fig. 30.13b), backpack blower (Fig. 30.13c), and backpack blower vacuum (Fig. 30.13d).



**Fig. 30.13** CAD models of products in the family. (a) Handheld blower, (b) handheld blower vacuum, (c) backpack blower, (d) backpack blower vacuum

### 30.7 Summary

The case study provides an example of using PPM (Chap. 8) to product family design. Although not all aspects of PPM were presented in the case study, the main concepts were discussed and demonstrated. This chapter began with defining the concept of the landscaping blower vacuum family to be designed. The customer demands were determined through direct and indirect interviewing, and the competing products from the Stihl and Echo companies. The product family was planned with offering four main market segment products and two niche products. Function structures for the six products in the product family were then created. Followed by modeling of the scaling platform components with the ability to accommodate the changes to the dimensions, the non-scaled platform components were also designed and modeled. Manufacturing and assembly improvements using the Lucas method was presented. The non-platform components are designed and modeled using modular architecture to provide the varieties using the platform.

The PPM design process for a product family focuses on multiple customer segments instead of one segment as normally found in a single product design process. Unlike an individual product design process and its homogeneous market segment, the case study demonstrates the use of four market segments to design the product family. When designing a product family, the design process needs to include plans for release dates for the product variants. In the case study, the plan includes the market release dates for initial four product offering, the two product niches and the upgrades to the models. In product family, economies of scale are

achieved through designing components to be leveraged across the family, known as a platform. The case study shows the use of an algorithm to identify and isolate components to select the platform. In product family, design components need is designed to be interchangeable to allow for product variants. In PPM, product variation is accomplished by modular architecture and customization of the variants to the different market segments or to the market niches within the main segments. PPM is used to design modular architecture and related connections for the lawn and landscape blower family.

## References

- Bower JL, Hout T (1988) Fast cycle capability for competitive power. *Harvard Bus Rev* 66:110–118
- Stalk GJ, Hout T (1990) *Competing against time*. The Free Press, New York, NY

# Epilogue

## Product Family and Product Platform Design: Looking Forward

**Timothy W. Simpson, Roger J. Jiao, Zahed Siddique,  
and Katja Hölttä-Otto**

The 30 chapters included in this book capture many of the recent advances in product family design and product platform development and encompass a broad scope of product fulfillment. Owing to the flurry of research activities, this field has matured rapidly in the past decade (Simpson et al. 2005; Jiao et al. 2007), and numerous industrial applications have involved product family and platform design (Simpson et al. 2006). These advances are motivating continued and renewed interests in design methods and tools to support the development of a product platform and the corresponding product family. While the chapters reflect the common themes and key developments in product family and platform design, it is important to achieve system-wide solutions for industries to deploy platform strategies. For future research, the following areas are identified as promising opportunities to advance the field.

---

T.W. Simpson ✉  
Mechanical and Nuclear Engineering, Penn State University, University Park, PA 16802, USA

Industrial and Manufacturing Engineering, Penn State University, University Park,  
PA 16802, USA  
email: [tws8@psu.edu](mailto:tws8@psu.edu)

R.J. Jiao  
Georgia Institute of Technology, Atlanta, GA, USA

Z. Siddique  
The University of Oklahoma, Norman, Oklahoma

K. Hölttä-Otto  
Engineering Product Development, Singapore University of Technology  
and Design, Singapore 138682 Singapore

## ***Customer Integration and Marketing Interaction with Product Family Design***

The driving force behind product family and platform design is the enterprise's positioning of customers at the center of value creation and involving customers into the product fulfillment process. Of primary importance in product families is the interaction with customers and marketing. On the technical side, designers have always assumed that customers' satisfaction with the designed product is sufficiently high as long as the product meets the prescribed technical specifications; however, what customers appreciate is not the enhancement of the solution capability but the functionality of the product. This means that the traditional dimensions of customer satisfaction may deserve scrutiny, for example, identifying product characteristics that cause different degrees of satisfaction among customers; understanding the interrelation between the buying process and product satisfaction; determining the optimal amount of customization and customer integration; explaining the key factors regarding the value perception of customers; and justifying an appropriate number of choices from the customers' and marketing perspective.

Equally important are customers' decision-making processes when interacting with product families and in turn developing proper fulfillment capabilities. Hence, it is important to support decisions of customers at the end, which coincides with consumer behaviors in business systems based on customer involvement in the product customization process. While most product family based customization approaches implemented in practice are based on offering a large variety of choices, the perception of choice and the joy or burden of configuration experienced by customers are not well understood (Chen et al. 2013). Many questions are pending. For example, what are the incentives for integrating customers into value creation? What factors drive customers to interact with a configurator? How many variants should be explored and offered before making a final decision? Are there any specific patterns that customers follow when interacting with a product family design system? And how do different levels of the decoupling point (i.e., where the customer is integrated into value creation) influence customer integration and how does this affect the performance of a product customization system? Towards this end, product family and platform development needs to be incorporated with more marketing-engineering decisions (Michalek et al. 2011; Williams et al. 2011) as well as customer perceptions and behavioral economies (Camerer et al. 2003; Koop and Johnson 2012).

In order to develop successful product families, companies must listen intently and identify the customer needs and expectations of each market segment and price/performance tier. In looking at this competitive landscape, each market niche needs to consider (Gordon 2004): What is the significance of this segment? What are the key products? What are their volumes, revenue, and profits? What is the outlook for the next 5 years? What does the company have to do to enter, sustain, and grow in the segment? The company needs to develop a comprehensive view of potential customers to understand their needs, requirements, and usage patterns. This voice-

of-the-customer (VOC) approach has been effective in helping guide the product specifications and features of new product platforms. Moreover, it is becoming more widely accepted that product family design approaches must be analytical and quantitative, that is, model based. One approach is to design product platforms for robustness, i.e., insensitive to variations (Simpson et al. 2006). It has been suggested that this can be best accomplished by using hierarchical and modular product architectures with appropriate interfaces to enable sensitivity analysis, error tracking, statistical analysis of uncertainties and their propagation, and cascading of requirements and specifications that enable both subcontractor flexibility and accountability (Otto 2005; Kokkolaras et al. 2006). It is interesting to observe that hierarchical frameworks are suggested for both traditional (physical product in engineering) and “nontraditional” applications (e.g., software engineering in Chap. 26).

### ***Corporate-Level Product Platform Support***

Platforms are related to the product architecture, supply chain, manufacturing, design reuse, etc. The platform strategy should be considered not only as a part of a product strategy but also as a corporate strategy (see Chap. 2 for more details). Platform design can be the tool to use to achieve diverse goals that align with the company strategy. The challenge is how to consider the full strategy in the development, i.e., how to take into account the multiple demands of the entire strategy while designing the platform.

In order to implement a broad and effective platform strategy, substantial management involvement is needed. Effective platform design requires a truly company-wide effort. A single platform should carry over through multiple product generations, but how many and how often should a single platform or the entire platform strategy be updated? Should a platform be adapted to changes when needed, or does that make the platform just a regular component that is redesigned as needed? Some researchers have addressed generational issues (Martin and Ishii 2002; Seepersad et al. 2002; and see examples in Chap. 29), but considerably more work is needed. The discipline of platform development represents a rich area for further research at the intersection of organizational behavior and engineering design.

While technologies such as Enterprise Resource Planning (ERP) and Product Data Management (PDM) have made inroads to supporting product development, there are currently few tools available to facilitate the sharing of knowledge directed to product platforms (Byron and Shooter 2005; Shooter et al. 2005). Opportunities abound for enhanced techniques for effectively capturing, storing, retrieving, and delivering information in support of product platform strategies. There is a need to explore how documents can become primary vehicles for manipulating an information model in support of platforms, implying the broader opportunities for knowledge management to support platforms. It is important for an organization to align with and sustain the platform strategy (Devendorf and

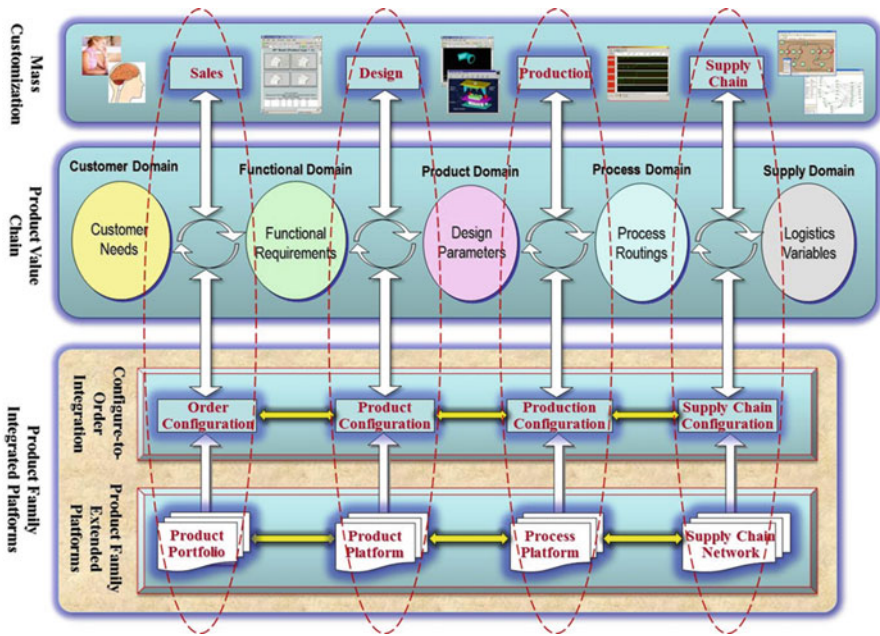


Fig. A.1 Extended platforms for comprehensive product families. Adapted from Jiao and Helander (2006)

Lewis 2011). The challenge is how to get support and involvement from the entire organization to this major change. There are clearly ample opportunities for research into organizations, operations, and human factors to support product platform strategies.

Although the basic principles of product family design are understood and well documented in the literature, quite a few fundamental issues require further examination. Some pending issues include, for example, the difference of customer-perceived variety from technical variety; the optimal degree of product differentiation; the mechanisms of interrelation between modularity and commonality; the implications of adaptability, flexibility, reusability, and customizability; the product family configuration models and decision support; and the coherence among the product architecture, family, and platform. Further issues may consider, for instance, to what extent a product family architecture and platform can best represent the capability of an enterprise? How can product families be matched with an existing set of resources and enterprise capabilities? How can various players (customers, designers, suppliers, production engineers, etc.) communicate well within the same platform of product family design? How to evolve product platforms and architectures in accordance with changes in customers' requirements, product technologies, and enterprise capabilities? How to coordinate basic product designs with variant design of the configuration process? Last but not least, what are the key factors that contribute to design-by-customer with the support of product families and e-commerce technologies?

## ***Extended Platforms for Comprehensive Product Families***

Product family design and development enhances profitability through a synergy of increased customer-perceived value and cost reduction in design, manufacturing, and the supply chain (see Chap. 23 for more examples of the latter). As discussed in Chap. 12, a product family should ideally be built on sharing a multidimensional core of assets such as standardized components, manufacturing, supply and distribution processes, customer segmentation, and brand positioning. To support coordination of the demand and supply chains with product families, it is necessary to extend platform thinking to the entire continuum of product fulfillment, including customer platforms, brand platforms, product platforms, process platforms, and logistics platforms, as illustrated in Fig. A.1. Greater complexity must be introduced to product family design decisions when considering more decision variables or design parameters pertinent to the coordination across the product, manufacturing process, and supply chain domains (Rungtusanatham and Forza 2005).

It is also important to incorporate the pervasive connectivity of the Internet to coordinate the participation of all parties including customers, suppliers, service providers, and many others. While the Internet facilitates a company's communication with its customers to configure its products and even offer online transactions, the manufacturers must implement such Web-based solutions to allow them to interactively communicate information related to product design, development, manufacturing, and logistics within their own infrastructures in a coherent manner. The ultimate goal is to achieve a virtual enterprise that enables digital product customization over the Internet through coherently integrating manufacturing production automation with supply chain management and sales-service support into a collaborative Web of interactive commerce.

The premise of product family design and platform-based product development relies on the belief that flexible manufacturing systems, along with scalable and modular product structures, can significantly reduce the fixed cost in comparison with the variable cost, enabling variety to be provided without incurring significant cost increases. Nonetheless, flexible manufacturing and the corresponding planning systems are necessary but insufficient for successful implementation of product families. These systems have to be supplemented by information technologies capable of handling the information flows and transaction costs involved in the fulfillment of product families. The advent of pervasive connectivity by the Internet provides the necessary and affordable connections among all parties in a product family-based customization system. As a result, the successful implementation of product families depends on the extension of concurrent engineering beyond the traditional boundary of design and manufacturing to include customer interaction, marketing, service, and recovery. It has been suggested that concurrent enterprising is very much in line with the idea of the real-time economy where the customers are central to the value creation. More research efforts are expected for building up rigorous frameworks of reconfigurable processes and process platforms, integrated information management for product and process families, coordination of product and process variety, etc.



Another important aspect of extended product platforms is for the enterprise to create capabilities with dynamic stability that enable the firm to adapt, integrate, and reconfigure the manufacturing skills and competences so as to react more sufficiently to new customers' requirements or to adapt to a changing business environment. Substantial research is needed to transfer the tools and principles from the design of this area to an enterprise that is no longer based on the manufacturing of products but on interactions with each individual customer through combinations of product-, software-, and/or service-oriented exchanges. It is necessary to establish methodologies for describing capabilities and sharing them in an extended value chain network. For example, how can modular process models be created and configured in order to integrate the capabilities of different firms in the fulfillment process of a specific customer order? The idea of interorganizational cooperation and virtual enterprises has to be developed much further. The benefits of integrating suppliers into customized manufacturing and capability development are well described in theory, but not well implemented in practice. Further research is needed to establish scalable and transferable coordination schemes in the logistics domain as well. Even though the trade-off of delayed satisfaction and increase in volume efficiency is obvious from the manufacturer perspective, it may not be the case for customers. This leads to the justification of the congruence of customer specifications and manufacturing capabilities, for example, to explain how major ERP suites (e.g., SAP, Baan, Edwards, etc.) affect operational performances when implementing product families.

### ***Financial Analysis of Product Platforms and Economics of Product Family Design***

Appropriate planning and architecture configuration for product family development requires estimating expected financial benefits both in terms of savings due to commonality (manufacturing, inventory, training, maintenance) and revenues due to successful product performance in the market (see examples in Chaps. 2 and 19). With only a few exceptions, most existing methodologies for product family design and development lack a rigorous cost-benefit analysis: cost models and data are either not available or proprietary, while expected product performance and revenues are estimated using elementary net present value methods. Most methodologies are based on the implicit assumption that maximized commonality is equivalent to maximized cost benefits. Even when cost models are included, they are used to quantify cost savings and to translate commonality to monetary units. Therefore, product commonality and differentiation is decided upon functional performance penalty (relative to products that do not share common parts or manufacturing processes) considerations without taking into account losses or profits due to market performance. Attempts to quantify the market impact of commonality in terms of demand and revenue effects and to "close the loop" with the manufacturing savings

(both fixed and variable costs) achieved through commonality are being made in both industry and academia. It is important to develop an end-to-end product modeling framework that maps key platform commonality decisions through both the product architecture to engineering performance and to product value-market revenue path and the product architecture to manufacturing cost and to investment finance path.

While the general causal relationships between key quantities such as commonality, product performance, market demand, revenue, and manufacturing costs are generally well understood, their detailed quantitative modeling remains elusive. The main reason is that as models of product performance, market demand, and manufacturing costs are concatenated, so too are the modeling errors and uncertainties inherent in them. To make matters worse, these errors are not typically additive but multiplicative. It is thus imperative to develop credible interdisciplinary product family development frameworks. Along with these, formal methods of model validation and verification against engineering, market, and cost data are needed. While the engineering part of the framework already exists and/or may not be the hardest one to achieve, the rest, such as integrated financial planning, is a challenge per se. As in single product development, in platform development too, the profits from the design take years to realize. The true success of a design can be objectively judged only at the end of the product's lifecycle. In platform development, this problem is even more pronounced as a platform is designed to last for several product generations. The question becomes how to evaluate the "goodness" of a platform sooner, rather during the development process already.

Product family design and development is indeed associated with new cost and profit structures that can entail "economies of scale and scope." Current research on the economic and performance evaluation of product families is dominated by empirical studies, ad hoc samples, or broad approaches based on cost accounting. Traditional cost accounting by allocating fixed costs and variable costs across multiple products may produce distorted cost-carrying figures due to possible sunk costs associated with investment into product and process platforms (Jiao and Zhang 2005). It is quite common in product family fulfillment that design and manufacturing admit resources, and thus the related costs, to be shared among multiple products in a reconfigurable fashion, as well as per-product fixed costs (Moore et al. 1999). Yano and Dobson (1998) observe a number of industrial settings, where a wide range of products are produced with very little incremental costs per se, or very high development costs are shared across broad product families, or fixed costs and variable costs change dramatically with product variety. They point out that "the accounting systems, whether traditional or activity based, do not support the separation of various cost elements." Safizadeh et al. (2000) derive similar results from an empirical study of 142 manufacturing plants: plants that provide a high degree of customization incur high-cost structures; however, when controlling for production processes, the trade-off disappears. This means once a company has defined its product range along with an appropriate production process, product family-based customization that falls within the range offered does not cost any extra.

The economic justification of product families requires the identification of proper measures and performance indicators to characterize different outcomes of a product customization system. This task is imperative because the current accounting systems are not designed for assessing the true economic benefits from the total value chain point of view (Cameron 2011). Even if the focus is shifted from cost control to value creation, existing accounting and control systems are mostly dominated by the practice of product costing. Savings and additional costs resulting from different degrees of interaction with the customers are not covered by most industrial accounting systems. Activity-based costing and the balanced score card approaches may provide initial solutions; however, approved ratios for calculating the value of customer relationships are still missing nor are parameters for evaluating the extent of the market research information gained by aggregated customer knowledge. Moreover, the value contribution of product families should be evaluated from the customers' perspective (see also Chap. 7). There is rarely any attempt to explicitly measure the need for individualization or to quantify the value of product families from a customers' perspective. The issue of justifying the economic value of introducing individualized products is of vital importance. Only if the increment in the customer-perceived value or utility suffices enough can product customization become a mass phenomenon. Recent study on the valuation of flexibility has suggested that the real option approach surmounts traditional discounted cash flow (DCF) analysis-based methods that tend to ignore the upside potentials from management flexibility (Kalligeros 2006; de Neufville and Scholtes 2011; Jiao 2012).

Furthermore, the risks related to product family development need to be addressed properly. Robertson and Ulrich (1998) observe the organizational risks related to platform development. Developing product platforms in most cases requires more investments and development time than developing a single product, which may delay the time to market and affect the return on investment time. Meyer and Dalal (2001) point out such risks, namely, that a weak common platform may undermine the competitiveness of the entire product line, and therefore a broad array of products may "feel the pain." In addition to fixed investments, developing platforms may result in over design of low-end product variants in order to enable reuse with high-end products (Krishnan and Gupta 2001). Henderson and Clark (1990) identify one potential negative effect of modular product architectures that originates from the risk of creating barriers to architectural innovation. Organizational forces may also hinder the ability to balance commonality and distinctiveness (Halman et al. 2003).

### ***Open Architecture Product and Service Platform Design***

Nowadays, there are so many new products being introduced to the market that they are no longer islands of their own to fulfill self-contained functionality. A modern product like an iPhone or iPad works not only because of its inherent industrial

and interface design but also because of the ecosystem in which it “lives” (Cho et al. 2010). Likewise, the recently launched MyFord Touch exemplifies a product-service ecosystem that has been designed to enable personalized in-car experience in the form of human interactions with the entire interior environment (Ford 2011). Along the same lines, a Xerox customer can choose to either purchase a printer with potentially a service plan, or simply a printing service, where the printer is located at the customer’s facility, but not owned by the customer anymore. These are all examples of a more holistic view of filling the customer need. As customers become more connected, products and services are increasingly knitted into a larger ecosystem of “touchpoints.” The physical product is not alone, whilst other factors can be conducive to the users’ emotional and hedonic experience, and in turn contributes to the value added (Gould 2010).

Services’ contribution to the world’s GDP has grown from 25 % in 1992 to 63 % in 2010 (World Bank 2012) while the percent contribution of manufacturing is declining. As a result, more and more companies from Rolls Royce to ZipCar are realizing that in order to increase revenues and profit, a new business model is required. This is resulting in blurred boundaries between products, services, and networks. It is often user experience that makes product-service ecosystems appealing in many industries (Rae 2006). We have been convinced by the trend of product value fulfillment progressing from traditional function-focused product and service fulfillment to nowadays customization and personalization (Jiao 2011). Pine and Gilmore (1999) have envisioned an “experience economy” underlying this paradigm shift, which has indeed come to fruition in many industries. Product and service design traditionally copes with physical products and emphasizes mainly functional requirements, yet with limited consideration of customers’ affective and cognitive needs or roles in decision making. It is therefore imperative for product and service design to bring in the human and system interaction dimensions explicitly (Papalambros 2010; Saunders et al. 2011). In particular, integrated design of product and service platforms suggests itself to be of paramount importance for achieving open architecture systems that enable open innovation, open design, and open manufacturing throughout the product realization process, so as to cater to business success in user experience.

Tremendous research opportunities are geared towards product and service family planning and open architecture platform development, and existing work has only “scratched the surface” (e.g., Moon et al. 2010). Mapping between the customer and functional domains constitutes the front-end issues associated with customization and personalization. Such a planning task usually starts with existing product portfolio and conforms to those common practices of order configuration and sales force automation. The exploration of “soft” user requirements involves intensive interactions with customers. Customer co-creation is necessary to elicit latent customer needs. User innovation, data mining, and Web learning lend themselves to be the main techniques of customer requirement acquisition and reasoning about user experience (Zhou et al. 2010).

Customization and personalization solutions are generated in the physical domain by mapping functional requirements to design parameters based on the

shared product and value chain platforms. The fulfillment of “hard” requirements involves typical decisions regarding product family design and configuration. For personalization of “soft” characteristics, customer-unique value chains must be designed in such a way that customer participation within a product-service ecosystem can be separated into a series of value-generating activities. Usability studies are always useful to design changeable and adaptable workflows that enable customer co-creation and accommodate open innovation. Also of concern are the cost advantages of personalized value chains. Similar to the wisdom of reusing proven design elements, formulating common value chain platforms is deemed to be an effective means to achieve mass production efficiency. New cyber-physical platforms, such as Web 2.0, cloud computing, P2P, and Second Life, offer great potential for implementing value chain platforms into online personalization engines that can provide recommendations on latent customer needs (Zhou et al. 2011).

The back-end issues associated with open architecture product and service platforms involve the process and logistics domains, which are characterized by process variables and logistics variables, respectively. The mappings from design parameters to processes and to logistics entail process platform design and supply chain platform planning. The main concern of process platform design is to take advantage of existing capabilities and utilize repetitions in production planning. The process view of personalization is enacted as service delivery processes. Identification of changeable, adaptable, and reconfigurable service delivery processes and formulation of service process platforms are deemed to be the fundamental issues of process reuse (Wiendahl et al. 2007). Likewise, in the logistics domain, the economic fulfillment of customization and personalization relies on changeable, adaptable, and reconfigurable supply and delivery networks.

Finally, the social aspect of product and service platforms is emerging as an interesting research area, as a product-service ecosystem is often associated with social networks (Jiao and Zhou 2013). Interactive information sharing among customers is becoming fast and convenient over the Internet with the online social networks (e.g., Facebook) or review sections of shopping websites (e.g., Amazon). The increasing availability of data about peer interactions and the popularity of marketing communication techniques based on such interactions have led to even greater interest in understanding the effects of peer influence on customers’ choice decisions of product offerings (Iyengar et al. 2011). The extensive reach of the Web and the prevalence of social networking sites have made large amounts of data on social networks easily available, which has recently resulted in their recognition as an important tool for marketing (van den Bulte and Wuyts 2007). Because the market is shifting to the online environment and due to the competitive nature of industries, it is important for firms to benefit from such information with appropriate marketing and product family design strategies (Panchal and Messer 2011). While the effect of peer influence has been well documented in marketing research (Childers and Rao 1992), social network effects have generally been neglected within the product family design process. Recent advances in social media that allow better access to social networks of customers have profound technical and

economic implications for product and service platform development. A phenomenal trend is emerging towards social commerce, which makes academia and industries recall the dot-com and e-commerce revolution of the previous decade ago. Abundant research opportunities exist in response to the emerging trend of open architecture product and service platform development that aims to leverage upon systems, humans, cybernetics, and businesses.

## *Summary*

Market pressures are forcing companies to rethink their product development organization, develop new technologies, infuse these into platforms, and derive customized variants from them. This encompasses the entire development process, from market and customer research to supply chain management. Integral to this change in a wide variety of industries is the adoption of a platform management architecture. Successful traits among industry leaders are the formation of cross-functional development teams, strong management support, common platform architectures that maximize the sharing of subsystems and components, and the ability to capture and apply lessons learned over time for continuous improvement.

Future research directions are aligned to the holistic view and system-wide solutions towards open architecture product and service platforms. More specifically, product family design needs to incorporate more front-end issues such as explicit customer modeling and integration, product demand, and market segmentation, along with the economic evaluation of product families. In the meantime, the spectrum of product family design should extend to include more back-end issues involving manufacturing, production, and the supply chain. The fulfillment of product families requires alignment of the customer, product, process, and supply chain decisions. Extended collaborative platforms are the core to support product customization over the Internet while achieving a synergy of sales force automation, product design, manufacturing planning, and supply chain management within a coherent framework.

It is important to think strategically about developing families of products or services and platforms based on scalable, modular architectures with interchangeable, standardized interfaces. In reality, complete modularity is not always fully achievable due to packaging, weight, power, and volume constraints, among others. Quantifying both the benefits and costs of platforming and standardization is necessary, but difficult due to inherent model and market uncertainties. It is these uncertainties that also require platforms to be designed with robustness or flexibility to respond to future needs better. These future needs could include new functional requirements demanded by customers, new technologies, social commerce, adherence to new regulations, or the expansion into new geographical and demographic markets. Product platforms tend to have lifetimes that exceed the lifetime of the variants that are derived from them, and this makes the problem both challenging and relevant.

# References

- Byron BM, Shooter SB (2005) A review of software solutions for the management of new product development and product family planning. In: ASME design engineering technical conferences, Long Beach, CA, Paper No. DETC2005/DAC-84454
- Camerer CF, Loewenstein G, Rabin M (2003) *Advances in behavioral economics*. Princeton University Press, Princeton, NJ
- Cameron BG (2011) Costing commonality: evaluating the impact of platform divergence on internal investment returns. Ph.D. Dissertation, Engineering Systems Division, MIT Press, Cambridge, MA
- Chen W, Hoyle C, Wassenaar H (2013) *Decision-based design: integrating consumer preferences into engineering design*. Springer, London
- Childers TL, Rao AR (1992) The influence of familial and peer based reference groups on consumer decisions. *J Consum Res* 19(2):198–211
- Cho CK, Kim YS, Lee WJ (2010) Economical, ecological and experience values for product-service systems. In: The 7th design and emotion conference, 4–7 Oct, Chicago, IL
- de Neufville R, Scholtes S (2011) *Flexibility in design*. MIT Press, Cambridge, MA
- Devendorf E, Lewis K (2011) The impact of process architecture on equilibrium stability in distributed design. *ASME J Mech Des* 133:101001-1–101001-12
- Ford (2011) <http://www.ford.com/technology/sync/myfordtouch>. Accessed 27 Jan 2011
- Gordon P (2004) Tapping the full potential of product platforms: best practices in planning, managing, and organizing for platform effectiveness. *Platform management for continued growth*. IIR/PDMA, Atlanta, GA
- Gould D (2010) Product ecosystems and service design. <http://www.psfk.com/2010/01/>
- Halman JIM, Hofer AP, van Wuuren W (2003) Platform-driven development of product families: linking theory with practice. *J Prod Innov Manag* 20(2):149–162
- Henderson RM, Clark KB (1990) Architecture innovation: the reconfiguration of existing product technologies and the failure of established firms. *Adm Sci Q* 35(1):9–30
- Iyengar R, van den Bulte C, Valente TW (2011) Opinion leadership and social contagion in new product diffusion. *Market Sci* 30(2):195–212
- Jiao RJ (2011) Prospect of design for mass customization and personalization. In: ASME international design engineering technical conferences and computers and information in engineering conference, DETC20011-48919, 28–31 Aug, Washington, DC, USA
- Jiao RJ (2012) Product platform flexibility planning by hybrid real options analysis. *IIE Trans* 44(6):431–445
- Jiao J, Helander M (2006) Development of an electronic configure-to-order platform for customized product development. *Comput Ind* 57(3):231–244
- Jiao J, Zhang Y (2005) Product portfolio planning with customer-engineering interaction. *IIE Trans* 37(9):801–814

- Jiao RJ, Zhou F (2013) Product line planning incorporating peer influence of social networks. In: IEEE international conference on industrial engineering and engineering management, Thailand
- Jiao RJ, Simpson TW, Siddique Z (2007) Product family design and platform-based product development: a state-of-the-art review. *J Intell Manuf* 18(1):5–29
- Kalligeros K (2006) Platforms and real options in large-scale engineering systems. Ph.D. thesis, MIT
- Kokkolaras M, Mourelatos ZP, Papalambros PY (2006) Design optimization of hierarchically decomposed multilevel system under uncertainty. *ASME J Mech Des* 128(2):503–508
- Koop GJ, Johnson JG (2012) The use of multiple reference points in risky decision making. *J Behav Decis Mak* 25(1):49–62
- Krishnan V, Gupta S (2001) Appropriateness and impact of platform-based product development. *Manage Sci* 47(1):52–68
- Martin MV, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. *Res Eng Des* 13(4):213–235
- Meyer MH, Dalal D (2001) Managing platform architectures and manufacturing processes for nonassembled products. *J Prod Innov Manag* 19(4):277–293
- Michalek JJ, Ebbes P, Adigüzel F, Feinberg FM, Papalambros PY (2011) Enhancing marketing with engineering: optimal product line design for heterogeneous markets. *Int J Res Market* 28(1):1–12
- Moon SK, Shu J, Simpson TW, Kumara SRT (2010) A module-based service model for mass customization: service family design. *IIE Trans* 43(3):153–163
- Moore WL, Louviere JJ, Verma R (1999) Using conjoint analysis to help design product platforms. *J Prod Innov Manag* 16(1):27–39
- Otto K (2005) Model-based design and verification of modular platforms. Innovations in product development conference—product families and platforms: from strategic innovation to implementation, Cambridge, MA
- Panchal JH, Messer M (2011) Extracting the structure of design information from collaborative tagging. *ASME J Comput Inf Sci Eng* 11(4):041007(1–11)
- Papalambros PY (2010) The human dimension. *ASME J Mech Eng* 132(5):1
- Pine BJ, Gilmore JH (1999) *The experience economy*. Harvard Business School Press, Boston, MA
- Rae J (2006) Ruthless focus on the customer. *Bus Week*, 28 Jul 2006
- Robertson D, Ulrich K (1998) Planning product platforms. *Sloan Manag Rev* 39(4):19–31
- Rungtusanatham M, Forza C (2005) Coordinating product design, process design, and supply chain design decisions, part A: topic motivation, performance implications, and article review process. *J Oper Manag* 23(3–4):257–265
- Safizadeh MH, Ritzman LP, Mallick D (2000) Revisiting alternative theoretical paradigms in manufacturing strategy. *Prod Oper Manag* 9(2):111–127
- Saunders MN, Seepersad CC, Hölttä-Otto K (2011) The characteristics of innovative, mechanical products. *ASME J Mech Des* 133(2):021009
- Seepersad CC, Mistree F, Allen JK (2002) A quantitative approach for designing multiple product platforms for an evolving portfolio of products. In: ASME design engineering technical conferences, Montreal, QC, ASME, Paper No. DETC2002/DAC-34096
- Shooter SB, Simpson TW, Kumara SRT, Stone RB, Terpenney JP (2005) Toward a multi-agent information infrastructure for product family planning and mass customization. *Int J Mass Custom* 1(1):134–155
- Simpson TW, Siddique Z, Jiao J (2005) Product platform and product family design: methods and applications. Springer, New York, NY
- Simpson TW, Marion TJ, de Weck O, Holttä-Otto K, Kokkolaras M, Shooter SB (2006) Platform-based design and development: current trends and needs in industry. In: ASME design engineering technical conferences, Philadelphia, PA, Paper No. DETC2006/DAC-99229



- van den Bulte C, Wuyts S (2007) Social networks and marketing, marketing science institute. Cambridge, MA
- Wiendahl HP, ElMaraghy HA, Nyhuis P, Zäh M, Wiendahl HH, Duffie N, Kolakowski M (2007) Changeable manufacturing: classification, design and operation. *Ann CIRP* 56(2):783–809
- Williams N, Kannan PK, Azarm S (2011) Retail channel structure impact on strategic engineering product design. *Manage Sci* 57(5):897–914
- World Bank (2012) <http://data.worldbank.org>. 10 Feb 2012
- Yano C, Dobson G (1998) Profit optimizing product line design, selection and pricing with manufacturing cost considerations, in product variety management: research advances. In: Ho T-H, Tang CS (Eds.), pp. 145–176, Kluwer Academic Publisher, Boston, MA
- Zhou F, Jiao RJ, Schaefer D, Chen S (2010) Hybrid association mining and refinement for affective mapping in emotional design. *ASME J Comput Inf Sci Eng* 10(3):031010-1–031010-0
- Zhou F, Xu Q, Jiao RJ (2011) Fundamentals of product ecosystem design for user experience. *Res Eng Des* 22(1):43–61

# Index

## A

ABC. *See* Activity-based costing (ABC)

Accommodation

- equal variability scheme, 567, 572–573
- performance, 560
- target user population, 560

Activity-based costing (ABC), 24, 480–481

Adaptive systems, 297, 307

Adjustability

- preferred seat locations, 1,000-member virtual population, 569
- sizing and the allocation, 582
- specification, 564–565

Aerodynamic particle separators

data mining-driven product family design, 167–168

engineering design optimization (*see* Engineering design)

market segments, 167, 168

methods, 167

particulate matter (PM)/particle pollution, 167

temporal market-driven preferences, 168–169

AHP. *See* Analytical hierarchy process (AHP)

AM. *See* Architecture model (AM)

Analytical hierarchy process (AHP), 694

Analytical network process (ANP)

consistency verification, 694

and QFD

consistency test, 703

construction, clusters and nodes, 702, 703

element, 703

matrix limitation, 703–704

weighted super matrix, 703, 704

Analytical target cascading (ATC), 275

ANP. *See* Analytical network process (ANP)

APFD model. *See* Associated product family design (APFD) model

Apple®, 100

Apple iOS, 748–749

Apple iPhone

computation, GVI values, 746

construction, QFD matrices

customer needs, 742

engineering metric target values, 742, 744

GVI process, 742, 743

design process and measures, 741

estimation, engineering metric target values, 744, 745

evolutionary and revolutionary changes, 741

generational variety index, 742

GVI

analysis, 747

matrix creation, 746

market life assessment, 742

normalized target value matrix, 745

predicting changes, customer needs, 745

user feedback, 741–742

Application Programming Interface (API), 653

Applied Research Laboratory's Trade Space Visualizer (ATSV), 458, 459, 469

Architecting

activity

clustering methods, 425

embodiments/building blocks, 426

function modeling, 425

knowledge base, computer tools, 427

software design, 426

system architects, 425

system-level specifications, 424–425

V-model, 424–425

- Architecting (*cont.*)
- design, software-intensive product
    - platforms (*see* Software-intensive product platforms)
  - process, product platforms
    - assets and modular platforms, 323
    - commonality assignment, 337–338
    - component-based approach, 332–333
    - customer needs gathering, 328–329
    - development, 325
    - functional requirements, 331
    - generic system, 333–334
    - market segment (*see* Market segments)
    - methods and techniques, 324–325
    - modules boundary (*see* Module)
    - module sizing and down selection, 338–339
    - roadmap, 336–337
    - system requirements, 329–331
- Architectural code generation, 661
- Architectural decomposition. *See* Decomposition
- Architecture
- map knowledge, 659–660
  - objects to design patterns mapping, 660
- Architecture-centric design approach
- AM, 422
  - architectural model, 422–423
  - complexity-related issues, 420
  - conceptual model, architectural description, 422, 423
  - connectivity information, 421
  - definition, 419–420
  - development activities, 423
  - domain-specific methods, 422
  - elements, product and system
    - architecture, 421
  - FBS model, 421–422
  - model views, product (platform)
    - development, 422
  - modularization and definition, interfaces, 431–437
  - platform designs, 423
  - platform development, 420
  - production context, 420
  - product platform development, 424–431
  - prototype tools, 445
  - stage of development, 445
  - structure of the product, 420
  - system architecture, 422
  - system architecture models, 437–446
- Architecture model (AM)
- architectural descriptions, 439–440
  - development activities, 440–441
  - entities “E” and functions “F”, 438, 439
  - FBS model, 422
  - information reuse and model-based paradigm, 441–445
  - information spectrum and design questions, 439
  - modeling conventions, objects and relations, 440
  - standard, 422, 423
- Assembly decomposition, 226
- Associated product family design (APFD) model
- algorithm, 85–87
  - cladistic analysis, 82
  - customer requirements, 87
  - demonstration and application, 87
  - and DSMs, 76
  - IDEF0 representation, 75
  - liaison graphs, 83
  - market segments and process plans, 75
  - parsimony analysis, 81
  - primary product components, 76–77
- ATC. *See* Analytical target cascading (ATC)
- Automotive
- and aerospace businesses, 127
  - industry, 135, 409
  - “small car platform,” enterprise, 122
- Axiomatic design, 26, 132, 504
- B**
- BACS. *See* Building air conditioning system (BACS)
- Benefits, commonality
- aircraft manufacturer, 58
  - broad firm cost structures, 58, 59
  - categories, 53, 54
  - comprehensive list, 54–56
  - cost saving, 54
  - design phase, 57
  - development cost, 59
  - firm’s flexibility, 54
  - management literature, 53
  - manufacturing cost, 59–60
  - manufacturing phase, 57
  - operation phases, 57, 58
  - proactive, 54
  - reuse, 54
  - works cited, 53
- Bicycle saddle height
- adjustability, 568–569
  - equal accommodation sizing, 572–573
  - equal variability sizing, 570–572
  - 42-member sample, 568

residual variance, 568  
 sizing, 569–570  
 Bill of material (BOM), 25, 26, 77, 120, 226  
 Black-box reuse profile, 555, 677  
 Body size and shape, 279, 560–562, 581, 582  
 BOM. *See* Bill of material (BOM)  
 Building air conditioning system (BACS)  
 Customer Intimacy, 111, 113  
 initial distribution, 112  
 The Middle East and Africa, 111  
 Operational Excellence, 111  
 Bulk purchasing, 60, 65, 66, 630, 643

## C

CAP. *See* Carryover Assignment Plan (CAP)  
 Carpal tunnel syndrome, 573  
 Carryover Assignment Plan (CAP), 257, 260  
 Carryover Chart (CoC), 258, 265  
 Case study  
 platform valuation  
 additional strategy cost, 190, 192  
 binomial lattice, 194  
 current and expected market segments, 192  
 design quality, 192–193  
 feature and component matrix, 190, 191  
 market analysis, 193  
 market segmentation, 190  
 mobile product family, 189  
 Nokia N70 series products, 189  
 option value, 194–196  
 platform design strategy, 189  
 tactile key marker, 190, 192  
 vision accessible, 189  
 Viper Case (*see* Viper Case study)  
 CCM. *See* Configurable Component Modeler (CCM)  
 CCs. *See* Configurable components (CCs)  
 CERs. *See* Cost estimating relationships (CERs)

CI. *See* Commonality index (CI)

## Cladistics

APFD model, 81  
 application, 72–73  
 biology, 73, 74  
 cladogram, 81–82  
 classification tool, 73  
 hypotheses, 73  
 modules identification and process  
 planning, 84  
 product modules, 82

## Cladogram

APFD model (*see* Associated product family design (APFD) model)  
 components, 82  
 construction process, 82  
 definition, 81  
 interspecies co-speciation, 74  
 product family, 83, 84  
 tanglegrams, 73

CMC. *See* Comprehensive metric for commonality (CMC)

CNC machine tools. *See* Computer numerically controlled (CNC) machine tools

CNs. *See* Customer needs (CNs)

CoC. *See* Carryover Chart (CoC)

Code division multiple access (CDMA), 658, 741

Code reuse evaluation, 675–676

Code reuse, product platforms vs. traditional approach

### application

dependent components, 675  
 independent components, 674–675

cost savings, 678–679

effects, 679–680

evaluation, 675–676

implementation, 673–674

methodology, 676–678

microelectromechanical system (MEMS) devices, 673

observations, 679

### PATF

black-box reuse, 677

compound reuse, 677, 678

product, 673

white-box reuse, 677, 678

## Commonality

assignment, architecting process, 337–338

benefits (*see* Benefits, commonality)

challenges, 33–34

characteristics, 12–14

CMC, CDI and TCI indexes, 14–16

### costs

design premium, 60

drawbacks and risks, 60, 61

individual variants, 62–63

investment, platforms, 62

multiproduct strategy, 60

realistic projections, 60–61

upfront variant, 62

and diversity, 26

and DSMs, 87

- Commonality (*cont.*)
  - and metrics, 12
  - modularity, 11–12
  - PCI, 11
  - and PFD, 74
  - PFEG, 25
  - product platform, 72, 73
  - selection and modularity, 18–20
  - vs. variety, family/platform design, 11, 25
- Commonality index (CI)
  - characteristics, 12, 13
  - definition, 12, 348
  - and measures, 73
  - and metrics, 12
  - standard, 318
- Commonality metrics
  - blocks, 304, 306
  - diagnostic value, 498
  - early-stage measures, 496
  - fabrication weighting, 498
  - investment-weighted metric, 498
  - IP beam comparison (*see* Instrument panel (IP) beam product families)
  - limitations, 498–500
  - modeling and assessment methods (*see* Modeling)
  - platforming decisions, 474
  - and platform literature, 475–476
  - shared components and type, product, 497
  - small-n case study, 474
  - total vs. fixed cost, 497
  - and variables, 303
- Commonality premium, 60, 62, 65
- Commonality selection
  - decision-making process, 466–467
  - design space exploration process, 469
  - engineering design practices, 469
  - interactive visualization methods (*see* Interactive visualization methods)
  - method 2, 467, 468
  - optimization, product family, 450
  - platform optimization approaches, 469
  - product complexity and competitive pressure, 450
  - product family design (*see* Product family design)
  - product optimization and visualization method, 469
  - product platform, 450
  - UTC product family, 467, 468
- Commonality strategy
  - canonical, 66
  - diffuse low-order, 66
  - realistic projections creation, costs, 60, 62
  - technical feasibility, 53
- Complexity
  - decomposition scheme, 288–290
  - encapsulation, 133, 134
  - robust optimization approaches, 34–35
- Component sharing
  - all-or-none restriction, 276
  - and commonality decisions, 26–27
  - end-of-life management, 714
  - optimization, 164
  - product family sharing level, 170
  - ranking, 497
  - ROI, 728
- Comprehensive metric for commonality (CMC)
  - CDI and TCI index, 15–16
  - commonality and variety, 25
  - commonality indices, 12–14
- Comprehensive product platform planning (CP<sup>3</sup>) model
  - application, electric motors family, 317–318
  - definition, generalized product platform, 305
  - design variables, 306
  - features, 300
  - formulation
    - commonality matrix blocks, 304
    - commonality variables, 302–303
    - design variable, 300–301
    - general product family, 302
    - modular product families, 303–304
    - modular-scaling families, 304
    - PFD, 301
    - product architecture, 305
  - generalized MINLP problem, 307
  - one-step approaches (*see* One-step approaches)
  - optimization process, 299
  - sample product family, 305
  - scale-based/module-based product families, 300
  - and SIO outcomes, 318–319
- Computer numerically controlled (CNC) machine tools
  - design, 599
  - fabrication and assembly tools, 598
  - machining setup elimination, 593–594
  - product/process design, 598
- Concurrent engineering, 27, 96, 781
- Configurable Component Modeler (CCM)
  - description, 137
  - FR and C objects, 138
  - PMC system, 140

- Configurable components (CCs)
  - CCM, 137, 138, 140
  - composition, 136–137
  - definition, 120, 130
  - and design rational (DR), 135, 138
  - PDM system, 138
  - system family, 135
  - and TEC, 139, 140
- Configuration
  - 2D, commonality chromosome, 278
  - design, 202
  - family, 27
  - platform and optimization (*see* Platform configuration and optimization)
  - platform and product family, 10–11
- Constraints and adaptability
  - code division multiple access (CDMA), 658
  - design knowledge, family product platform, 657
  - dual-tone multi-frequency (DTMF) generator, 658
  - flow balance, product lifecycle components, 718–719
    - core availability, 720
    - cores, 716–717
    - demand satisfaction and avoidance, 720–721
    - environmental regulations, 720
    - intermediates, 717
    - refurbished cores, 719–720
    - refurbished intermediates, 719
    - variable condition, 721
    - working and nonworking cores, 716–717
  - network protocol, 658
  - social environment changes, 659
- Construction equipment accessory
  - modular function deployment, 114
  - module indication matrix, 114
  - practice, 115
  - “strategic suppliers”, 114
  - value disciplines, 114–115
- Control devices, industrial trucks
  - CoC, 265
  - modular product programs, 263
  - module variants, 265–266
  - VAM, 264, 265
- Cordless handheld vacuum
  - conceptual modules, 617, 620
  - CVR (*see* Customer value rating (CVR))
  - description, 610–611
  - DPM, 614, 616
  - exploded view, 12 V unit, 610, 612
  - market segments (*see* Market segments)
  - MIM, 616–617
  - module generation, 617, 618
  - QFD, 614, 615
  - specifications, 611–612
- Cost estimating relationships (CERs), 633
- Cost modeling
  - ABC, 480–481
  - cost savings metric calculation, 483
  - exclusive cost, 483
  - fixed costs, 403
  - PBCM, 481–482
  - process-based, 481
  - standalone cost, variant, 482–483
  - total cost and profit, 405
  - variable, 404
  - volume-weighted ratio, 483
- CP<sup>3</sup> model. *See* Comprehensive product platform planning (CP<sup>3</sup>) model
- Customer needs (CNs)
  - architecting process, product platforms market segmentation, 328, 329
  - planned market strategy, UGVs, 328
  - requirements, 328
  - voice of the customer (VOC), 328–329
  - description, 504
  - and FRs, 504
  - GVS description, 525
  - PPM
    - direct interview methods, 205
    - indirect feedback method, 206
    - market segment, 204–205
  - product family sizing design (*see* Product family sizing design)
- Customer preference
  - clustering, 7
  - involvement, 8
  - and market conditions, 34
  - QFD, 149
- Customer satisfaction
  - competitive and dynamic marketplaces, 344
  - product profiles, 353–354, 359, 362
- Customer value rating (CVR)
  - modular product, 606
  - and QFD, 616
  - target market segments, 614
- Customer values
  - benefits, 608
  - QFD, 614
  - transformation, product properties, 606
- Customization
  - appearance, 10
  - family-based system, 781
  - mass, 2, 25, 148, 544, 600–601

- Customization (*cont.*)  
 and personalization, 785, 786  
 platform-based, 201–202  
 product families, mass, 601–602  
 CVR. *See* Customer value rating (CVR)
- D**
- Data mining  
 description, 154  
 iterative feature evaluation, 154–156  
 KDD process, 154  
 model generation and irrelevant feature  
 classification, 156–158  
 nonstandard feature (NF), 159–160  
 obsolete feature (OF), 160–161  
 standard feature (SF), 158–159
- DCDM. *See* Design of commonality and  
 diversity method (DCDM)
- DCF. *See* Discounted cash flow (DCF)
- DCI. *See* Degree of commonality index (DCI)
- Decision-making  
 collaborative, 182  
 commonality, 53, 452–453, 466, 467  
 complexity, 465  
 cultural, 53–54  
 customers', 780  
 enterprise, 147, 174  
 market-based, 180  
 strategic, 394, 395
- Decision support to design process planning  
 construction, TCPN, 520  
 TCPN deployment, 520
- Decomposition  
 assembly, 226  
 ATC, 275  
 commonality, 221–222  
 description, 221  
 functional, 226  
 granularity (*see* Granularity)  
 hierarchical decomposition, 224  
 and parallelization, MOGA  
 chromosome representations, 280, 281  
 commonality value, 282  
 crossover and mutation, 282  
 description, 282–283  
 fitness calculation, 280, 282  
 initialization, 280  
 iteration and termination, 282  
 replacement, 282  
 products and systems, 224  
 representation  
 DSM, 222–223  
 functional model, 222, 223  
 network, 222, 223  
 service-based, 227  
 top-down decomposition, 224  
 viewpoint, 225–226  
 work breakdown structure, 224
- Degree of commonality index (DCI), 150
- Dendrogram, 617, 618
- Dependency matrix, 27
- Design concept exploration, representative  
 solutions  
 cluster 1, 412, 413  
 cluster 2, 412, 413  
 cluster 3, 413, 414  
 cluster 4, 413, 414  
 cluster 5, 414, 415  
 cluster 6, 414, 416  
 cluster 7, 415, 416  
 cluster 8, 415, 416
- Design decision-making, 33, 739
- Design for Human Variability (DfHV)  
 adjustability, 564–565  
 bicycle saddle height (*see* Bicycle saddle  
 height)  
 description, 559, 560  
 multiple sizes, 576–577  
 opportunities, platforming and  
 modularity, 577  
 platforming and modularity, 577–582  
 principles, 560–561  
 quantifying, variability  
 anthropometry synthesis, 562–564  
 databases, 561–562  
 robust design methodologies, 559–560  
 sizing (*see* Sizing)  
 tool handle  
 carpal tunnel syndrome, 573  
 grip diameters, 575  
 optimal diameter, 574  
 regression, 574–575  
 RMSE, 575  
 single size, 575–576  
 user's grip circumference, 574
- Design for lifecycle, 54, 783
- Design for manufacturability (DFM)  
 and assembly, 99  
 guidelines, 600  
 platform, 211, 213  
 principle, 594, 599
- Design for recovery  
 cost and revenue parameters, 724–726  
 end-of-life recovery, 711–714  
 optimization problem, 714–715  
 product take-back, 710–711
- Design for variety (DFV)  
 definition, 72  
 product variety, 253

- TEV, 252–253
- VAM, 254
- Designing product families
  - arbitrary decisions, 600
  - CNC, 599
  - DFM principles, 599
  - eliminate tool, 599
  - manufacturability, 600
  - products development, 598
  - standard parts and materials, 599–600
- Design matrix
  - DPM (*see* Design property matrix (DPM))
  - DSM (*see* Design structure matrix (DSM))
- Design of commonality and diversity method (DCDM), 181
- Design process
  - actor and resource allocation, 527, 529
  - characteristics, 505
  - CNs, 504
  - configurations, TCPN, 528
  - customers' requirements, 504
  - Gantt chart, 527, 528
  - generic structure, 506
  - GVS, PNs and TCPN model, 505
  - modular design projects, 506–507
  - modularization (*see* Modularization)
  - performance, 527, 529
  - planning, 525–527
  - platform-based techniques, 528
  - PN (*see* Petri Nets (PN))
  - product variant (*see* Product variants)
  - project module identification (*see* Project module identification)
  - TCPN, design process simulation, 525, 526
- Design process planning, 519–520, 525–527
- Design project
  - configurable component-based product platform model, 131
  - modular design, 506–507
- Design property matrix (DPM)
  - company-specific requirements, 610
  - description, 614, 616
  - hierarchical clustering, 617
  - MFD, 607
  - and MIM, 607
  - module generation, 617
  - and module indication matrix (MIM), 96, 606
  - physical embodiments of functions, 610
  - and PMM, 606, 607
  - product architecture, 607
  - technical solutions, 610
  - voice of engineering, 614
- Design quality
  - customers' preference, 187
  - expected, 192
  - family product, 186
  - full quality, 187, 192
  - functions, 187, 188
  - marginal quality, 187, 192
  - market demands, 188
  - N71, 192–193
  - value of the preference  $U(Q_p)$ , 188
- Design rationale (DR)
  - CC models, 138–140
  - concepts, 131
  - configurable component, 135–136
  - and design histories, 120
  - system description, 130
- Design structure matrix (DSM)
  - analytical methods, 507
  - APFD model, 76, 77, 80
  - application, 87
  - architecture, 222–223
  - assembly decomposition, 226
  - automatic naming algorithm, 432, 433
  - based modularization, 434
  - clustering techniques, 434
  - component-based, 333
  - description, 248, 434, 511
  - DMM, 434
  - and DMMs, 511
  - FBS model, 436
  - “generic” architecture, 333–334
  - herbicide spraying systems, 249
  - heuristic swapping algorithm, 507
  - idealized matrices, granularity, 229–230
  - interactions, 164
  - kettles product family (*see* Kettles family)
  - market segments, 80
  - market segments and kettles primary components, 77, 79
  - mathematical operations, 507
  - matrix operations, 511
  - off-diagonal entries, 434
  - paper trays, 235
  - primary and secondary components in kettles, 77, 80
  - project elements, 522
  - service-based decomposition, 235, 240, 241
  - symmetrical and asymmetrical, 223
  - and UGVs
    - clustered, 335, 336
    - unclustered, 334–335
  - viewpoints, printing system, 235–237



- Design valuation
    - company's profit, 184–185
    - design quality, 187–188
    - financial model, 185–187
    - product family architecture, 183
  - Design versatility, 599, 602
  - Development cost, 59
  - DfHV. *See* Design for Human Variability (DfHV)
  - DFM. *See* Design for manufacturability (DFM)
  - DFV. *See* Design for variety (DFV)
  - Disassembly
    - constraints, 716–719
    - description, 709
    - end-of-life recovery, 711–714
    - system decomposition, 226
    - yield rates, 723–724
  - Discounted cash flow (DCF), 784
  - Divergence
    - commonality, 50
    - planning
      - beneficial, 63
      - commonality levels, 63
      - firm's ability, 65
      - program manager, 65–66
      - rail manufacturer, 65
      - realistic commonality benefits, 65
      - research data categorization, 63, 64
      - variants, 63
  - DMMs. *See* Domain mapping matrices (DMMs)
  - Domain analysis
    - architectural code generation, 661
    - architecture objects to design
      - patterns, 660
    - constraints and adaptability, 657–659
    - detailed software design, 660–661
    - domain knowledge model, 656–657
    - high-level process, product family platform, 654
    - knowledge to architecture, 659–660
    - previous product projects, 655
    - relevant literature, 655
    - use cases, 654–655
  - Domain knowledge model, 656–657
  - Domain mapping matrices (DMMs)
    - and MoAs, 510
    - modularization process, 434
    - project elements, 522
    - types, relationships, 511
  - DPM. *See* Design property matrix (DPM)
  - DSM. *See* Design structure matrix (DSM)
- E**
- Economies of scale
    - advantages, 451
    - and customization, 119, 120
    - description, 57
    - product family design, 181
    - in recovery operations, 733
  - End-of-life management
    - benefits, 710
    - definition, 709
    - hierarchical assembly structure, product family, 709
    - product take-back, 710–711
    - recovery (*see* End-of-life recovery)
  - End-of-life recovery
    - disassembly parts and decision making, 713, 714
    - disposal and recycling, 712
    - incoming feedstock, 711
    - options, 711–712
    - process, 712, 713
    - refurbishment/cannibalization, 713–714
  - Engineering characteristics (EC)
    - platform and non-platform (*see* Optimization, platform and non-platform ECs)
    - scalable product platform, 345
    - sensitivity analysis, 351–352
    - value determination, 350–351
  - Engineering design
    - aerodynamic
      - design objectives, 170
      - mapping product, 174
      - mathematical representation, 170–171
      - particle separator design, 169
      - product family sharing level, 170
      - scale based and module based, 169–170
    - and PFD
      - commonality, 162
      - component function identification, 162, 163
      - cosine similarity, 164
      - data mining predictive model, 161–162, 173–174
      - DSM, 164
      - large-scale data set, 162
      - latent semantic analysis, 162
      - product family optimization problem, 165
      - product feature-function comparison matrix, 162–164
      - product variant optimization level, 166–167

- sharing level, product family, 166
  - textual description, 162
  - 60 Whr 6-Cell Lithium-Ion Battery, 164–165
- Ergonomics, 560, 562
- Evolutionary optimization
  - optimization strategy, 634
  - three-objective discrete optimization process, 637–638
  - two-objective method, 634–637
- F**
- FBS modeling. *See* Function-behavior-state (FBS) modeling
- Financial model
  - call option valuation, binomial tree, 187
  - demand, movements, 185–186
  - design strategies, 196
  - drift, 185
  - family product, 186
  - modular product families, 196
  - net benefit, 186
  - real options analysis, 180
  - time interval, 186
  - valuation, 182
- Fit, 590
- Flexible assembly, 595
- Flexible building, product families
  - assembly, 595
  - CNC, eliminate machining setup, 593–594
  - flow manufacturing, 594
  - one-piece flow quality, 594
  - setup and batch elimination, 593
  - source cells, 594
  - tooling setup elimination, 593
- Flexible platform, 3, 5, 14, 25, 31, 748
- F-M tree. *See* Function-means (F-M) tree
- FRs. *See* Functional requirement (FRs)
- Functional decomposition, 226–227
- Functional efficiency, 213, 217
- Functional requirement (FRs), 504
- Function-behavior-state (FBS) modeling
  - DSM, 434
  - product architecture, 421–422
  - shifting system, 434
  - system architect, 432
- Function-means (F-M) tree, 132
- Function strategy, 208
- Function structure
  - component design modeling, 214–215
  - elements, 763–764
  - function strategy, 208
  - handheld blower, 761, 764, 765
  - modularization, 686
  - modules, 335
- Fuzzy clustering
  - combined similarity measure, 514
  - $D_0$ ,  $D_1$ , and  $D_2$ , 513
  - equivalence matrix, 514
  - hierarchical decomposition, 511
  - module identification, 511, 512
  - partitioning algorithm, 512
  - project elements, 512
- Fuzzy optimality, 347, 628
- G**
- Gas inlet valves
  - MIG, 261, 263
  - MPC, 255, 261, 262
  - product variants, 263
  - TEV, 261–262
  - vacuum applications, 261
  - VAM, 261, 262
- GBOMs. *See* Generic bill of materials (GBOMs)
- Generalized commonality. *See* Joint product platform selection and family design
- Generational variety index (GVI)
  - analysis, 747–748
  - Apple iPhone (*see* Apple iPhone)
  - calculation, 740, 746
  - customer requirement list, 743, 746
  - development, 749
  - evolving designs
    - consumer electronics segment, 738–739
    - impacts, 739
    - platformed and non-platformed products, 739
    - product features, 738
    - revolutionary/evolutionary changes, 738
  - iOS-based flexible platform strategy, 748–749
  - limitations, 749
  - low and high values, 337
  - Pareto optimal designs, 741
  - product platform design, 740–741
  - risk, 748–749
  - technological advancement, 738
- Generic assets
  - bandwidth, 124, 127
  - bill of material (BOM), 120
  - description, 144
  - development processes and concept platform, 125–126
  - industrial case study

- Generic assets (*cont.*)
- CCM software, 140
  - commercial software RD&T, 140, 142
  - design rationales (DRs), 139
  - design solution space, 142
  - fabricated structures, 139
  - functional requirement-related (FR) properties, 140
  - parameter values, 142
  - platform elements, 140
  - TEC family system (*see* Turbine exhaust cases (TEC))
  - industrial customers, 125
  - knowledge platform approach, 124, 125
  - life cycle phases, 143
  - maturity, 123–124
  - OEM company, 125
  - platform-based development and manufacturing, 119–120
  - platforms (*see* Platform)
  - PLM (*see* Product life cycle management (PLM))
  - product and production system platforms (*see* Product, and production system platforms)
  - reuse, 120–121
  - suppliers, 121
  - system development project, 126–127
  - technology platforms, 127–129
- Generic bill of materials (GBOMs), 20, 506
- Generic product plan, 208, 209
- Generic variety structure (GVS)
- hierarchical structure, 509
  - leaf node activities, 509
  - process varieties, 509, 510
  - AND and XOR junction, 510
- Genetic algorithm (GA)
- GP, 23
  - GVS, 514
  - MINLP, 316
  - parametric platforms, 4
  - Pareto optimal solutions, 394
  - product family design (PFD) method, 296
- Genetic programming (GP), 23
- Global product family design
- automotive industries, 409
  - availability, modules, 409, 410
  - clustering, Pareto solutions, 411
  - demand volume, products, 409, 410
  - design concept exploration (*see* Design concept exploration, representative solutions)
  - global manufacturing, 394–395
  - mass customization, 393
  - mathematical model, 396–404
  - multi-objective
    - and concept-level design, 395–396
    - formulation, 416
    - optimization, 410–411
  - neighborhood cultivation genetic algorithm, 416
  - optimal techniques, 407–408
  - PCA-based clustering technique, 416–417
  - price, products, 409, 410
  - production and sales capability, respective sites, 409, 410
  - site differences, 409, 410
- Granularity
- assembly modules, 238
  - DADF and xerographics modules, 233–234
  - degree of modularity
    - $M_{G\&G}$  metric, 238–239
    - service-based decomposition, 239–241
  - description, 241–242
  - DSM (*see* design structure matrix (DSM))
  - effect, modularity
    - idealized matrices, 230, 231
    - integral and modular variants, 230, 231
    - MDL, 230, 232
    - $M_{G\&G}$ , 230, 232
    - product, 229
  - electrical components, 237–238
  - ITB module, 234, 235
  - metrics, 228–229
  - printing system, 235
  - service-based decomposition, 235
  - “smaller size”, 236
  - system decomposition, 233
  - Xerox DocuColor 250, 233, 234
- GVI. *See* Generational variety index (GVI)
- GVS. *See* Generic variety structure (GVS)
- H**
- Hierarchical clustering
- ECs, 351–352
  - MFD, 606, 617
  - modules, 334
  - multidisciplinary dependency, 74, 75
  - SPSS, 617
- HoQs. *See* Houses of quality (HoQs)
- Houses of quality (HoQs), 357
- Human factors, 560, 562, 780
- Human variability. *See* Design for Human Variability (DfHV)

**I**

INCOSE guidelines, 329

Industrial machines, software platforms  
 changing design  
   family platform framework layer, 667–668  
   final product test, 669–671  
   product family infrastructure layer, 666–667  
   product-specific features layer, 668–669  
 description, 662  
 high-level organization  
   activedevices, 663–664  
   PATF engine and component, 662–663  
 modularity, 667  
 review, 667  
 run-time system composition  
   application sequences, PATF, 671–672  
   AppProductSequence object, 673  
   object composition, 671  
   SyncPoint objects, 673  
 system behavior, high-level  
   automatic mode, 665–666  
   manual mode, 664–665  
   PATF system state, 664, 665

Instrument panel (IP) beam product families  
 assembly cost allocation, 486  
 beam processing information, 484, 485  
 commonality metrics  
   component and magnesium, 484, 487  
   correlations, 494, 495  
   and cost savings, 491, 493  
 investment-weighted commonality metric, 491–492  
 linear regression analysis, 494  
 magnesium design, 484, 485  
 mass-and piece-weighted metrics, 494  
 operational and financial assumptions, 484, 485  
 and PBCM, 484  
 product variants, 484, 486  
 projected cost category, 496  
 proposed commonality metrics, 486  
 regression statistics, 494–496  
 standalone variants  
   assembly costs, 484, 491  
   development costs, 484, 491  
   fabrication costs, 484, 491  
 steel and magnesium IP beam variants, 486, 492  
 steel component and commonality information, 484, 488–490  
 volume-weighted commonality metric, 491

Interactive visualization methods  
 exhaustive + visualization  
   design space exploration, 464  
   product family design, 456, 457  
 individual opt + visualization  
   ATSV, 458  
   commonality selection, 459–461  
   multidimensional data, 459  
   product family design space, 459  
   trade-off resolution, 464–465  
 Pareto Band concept, 466  
 product family opt + visualization  
   CI, 462  
   commonality assessments, 463  
   computational cost, decomposition, 466  
   family-based optimization techniques, 461  
   product family optimization, 462  
   trade-off resolution, 465  
   UTC product family, 461

Interface definition  
 DSM, 434  
 FBS model, shifting system, 434–437  
 and modularization, 431  
 workflow and function modeling, 431–434

Intermediate transfer belt (ITB) module, 234, 235

Inventory reduction, 55, 56, 58

iPad<sup>®</sup>, 101, 749, 784–785

IP beam product families. *See* Instrument panel (IP) beam product families

iPhone evolution. *See* Generational variety index (GVI)

ITB module. *See* Intermediate transfer belt (ITB) module

**J**

Joint product platform selection and family design  
 classification, product family optimization, 273–274  
 decomposition approaches, 275  
 issues, 272  
 MOGA (*see* Multi-objective genetic algorithms (MOGA))  
 optimization-based research, 272  
 pareto fronts  
   electric motor family, 287  
   GAA family, 286  
 platform configuration, 287  
 posteriori optimization methods, 274  
 product platforms, 272

Joint product platform selection and family design (*cont.*)  
 single-stage class III problems, 274  
 universal electric motor, 283–291  
 Joint Strike Fighter, 34, 49, 50, 63

## K

Kanban, 594–597  
 Kettles family  
 liaison graph, 83  
 market segments, 76, 80, 86  
 primary components, 77, 79  
 process plan, 87  
 secondary component, 77, 80  
 unique feasible kettles variants, 80, 81  
 water boiling, 76, 77  
 K-means clustering, 351, 690, 692–693, 695  
 Knowledge discovery in databases (KDD)  
 acquisition, 152–153  
 description, 151–152  
 mining/pattern discovery (*see* Data mining)  
 selection and cleaning, 153  
 transformation, 154  
 Knowledge to architecture mapping, 659–660

## L

Latent semantic analysis (LSA), 162, 164  
 Lawn and landscape blower family, PPM  
 business environment, 753  
 data, customer needs, 754  
 formation strategy, 753–754, 762  
 function structure  
 engine control system, 761, 763  
 handheld blower market target, 761, 764, 765  
 main blower process system, 763  
 market research, PPM (*see* Market research, PPM)  
 modularity construction (*see* Modularity construction, PPM)  
 platform design  
 CAD model components, 768–769  
 components list, platform finding algorithm, 765–766  
 feeding analysis, 769–771  
 fitting ratio, 771  
 handheld blower vacuum, 764, 765  
 input sheet, 764  
 manufacturing analysis, 771, 772  
 outcomes, 766–767  
 permanent components, 767

product family planning, 761, 762  
 product variations, 753, 754  
 resources, 754–755  
 road map, 753, 754  
 rules and regulations, 755

## Learning curve

bulk purchasing and construction, 643  
 manufacturing line, 57  
 typical ship construction labor, 630, 631

## Liaison graph

APFD model, 83  
 application, 83, 87  
 definition, 83  
 kettles, 83

## Life phases modularization

module drivers, 255–256  
 MPC, 255, 256  
 product life cycle, 254–255

## LSA. *See* Latent semantic analysis (LSA)

Lucas method, 211, 213, 215, 218

## M

### Mann–Kendall (MK) trend test

feature  $F_i$ , 157  
 feature  $F_n$ , 159  
 feature  $F_o$ , 160–161  
 feature  $F_s$ , 158–159  
 mathematical representation, 157–158  
 null hypothesis, 158

### Manufacturing cost, 59–60

### Market-based design, 180–182

### Market-driven product family design (MPED), 346

### Market mechanisms, 28, 180

### Market research

competitors, 206–207  
 customer needs, 204–206

### PPM

competitor product analysis, 758–759  
 competitors market targets vs. customer requirements, 759–760  
 customer demands, 759, 760  
 customer fulfillment, 758–759  
 direct and non-direct polling, customer demands, 755–757  
 identification, market segment, 755, 756  
 market target, 757–759  
 rules and regulation bodies, 207  
 volume analysis, 207

### Market segments

and APFD model, 75  
 business application, 325

- characteristics, 325
- current and expected, 192
- customer
  - population, 325
  - preferences, 187
  - requirements, 74
- defense-related applications, 325
- demand model, 182
- description, 611
- difference, 325
- and DSM, 77, 80
- family Felicia, 613
- functional preference information, 187
- identification, 180
- kettle identification, 76, 86
- and market attack plan, 326–327
- mobile products, 190, 192
- platform leveraging strategies, 327
- preference value and demographics, 188
- product market matrix, 327–328
- scalable platforms, 328
- Sophia student, 613–614
- sources, 76
- suction power, 613
- UGVs, 326–327
- Mass customization
  - extending product families, 601
  - postponement, 600–601
  - setup “elimination”, 593
  - synergies, 601–602
- Mass production
  - cost-oriented design method, 246
  - product families, 591–592
- Mathematical model
  - conditions, problem formulation, 397–398
  - cost model, 403–405
  - delivery model, 405–406
  - integration, product family and supply chain, 396–397
  - mathematical formulation, 406–407
  - product family model, 399–402
  - quality model, 405
- MDL. *See* Minimum description length (MDL)
- MDO. *See* Multidisciplinary design optimization (MDO)
- MEMS. *See* Microelectromechanical system (MEMS)
- Method units, integrated PKT-approach
  - description, 251
  - design for variety, 252–254
  - life phases modularization, 254–256
  - modular product programs, 257–258
  - product program planning, 256–257
  - visual tools, 252, 253
- Metrics
  - activity-based costing system (ABC), 24
  - assessment methodology, 483
  - axiomatic design, 26
  - commonality (*see* Commonality)
  - correlations and dependencies, 26–27
  - cost considerations, 21, 24
  - cost model, 25
  - definition, 24–25
  - extended QFD, 25
  - Ford’s Model T, 26
  - fuzzy clustering techniques, 26
  - leveraging data mining techniques, 25
  - multi-criteria platform evaluation, 24
  - platform construction method, 25
  - platform exploration and identification, 26
  - product family evaluation graph approach (PFEG), 25
    - variation and uncertainty, 25, 26
- MFD. *See* Modular function deployment (MFD)
- Microelectromechanical system (MEMS), 673
- MIG. *See* Module Interface Graph (MIG)
- MIM. *See* Module indication matrix (MIM)
- Minimum description length (MDL), 228, 232, 239, 240
- Mission effectiveness, 627, 629–630
- Mixed-integer nonlinear programming (MINLP), 300, 301, 307, 316, 454
- Mixed integer programming, 714, 733
- MK trend test. *See* Mann–Kendall (MK) trend test
- MoAs. *See* Module of activities (MoAs)
- Modeling
  - cost methodology (*see* Cost modeling)
  - metric assessment methodology, 483
  - product family, 9–10
  - proposed commonality metrics
    - bill of materials, product family, 478
    - calculation, 478, 479
    - descriptions, 480
    - metric and cost savings, 477
    - piece-based, 478
    - production volume, 479
    - subassemblies, 478
    - trimming/drilling, holes, 477
- Modular function deployment (MFD)
  - and BACS (*see* Building air conditioning system (BACS))
  - business strategy, 117

- Modular function deployment (MFD) (*cont.*)
- cell phone, 115–117
  - complexity cost reduction, 605
  - construction equipment accessory
    - (*see* Construction equipment accessory)
  - cordless handheld vacuum (*see* Cordless handheld vacuum)
  - cross-functional, 94
  - design property matrix, 609–610
  - and DFX approaches, 96
  - documentation and analysis, 607
  - DPM and MIM, 607
  - functional requirements, 94–95
  - indication matrix, 610
  - industrial companies, 117
  - interfaces analysis, 96
  - launch planning (*see* Modular launch planning)
  - matrix-based method, 605
  - and MIM, 96
  - module drivers (*see* Module drivers)
  - NAICS, 106
  - and PMM, 94, 95, 606, 607
  - product family and development, 91, 92
  - product properties, 94
  - and QFD, 94, 95, 608–609
  - riding-machine platform (*see* Riding-machine platform)
  - strategy and tactics, 92–93
  - tactical vehicle, 93–94
  - unique application, 106
  - “Voices of X”, 94
- Modularity
- cohesion and coupling, 541–542
  - commonality, 11–12
  - goals, 540–541
  - highly coupled structure, 541
  - PFD analysis
    - APFD model algorithm, 85–87
    - cladistics, 81–82
    - product modules and platforms, 82–83
  - and platform
    - long-lifetime products and secular trends, 580–582
    - long-tailed and skewed distributions, 577–579
    - segmented populations and disproportionate disaccommodation, 579–580
  - principle of separation, 540
  - SMI, 24–25
  - software production, 540
- Modularity construction
- component design modeling, 214–216
  - component manufacturing tooling
    - design, 215
  - Lucas method, 215, 217
  - market targets, 215
  - module design, components, 214
  - PPM
    - CAD model, 771, 773–775
    - design, 771, 774
    - platform and non-platform components, 771
    - products, 774–775
    - production volume costing, 215, 217–218
    - types, 203
- Modularization
- actor and resource, dashboard design, 523, 524
  - car dashboard design, 521
  - design activities, 522, 523
  - DSM representation, 522
  - fuzzy clustering algorithm and MoAs, 522
  - generic design process, 522, 523
  - generic routing, dashboard design process, 523, 524
  - input/output class, 523, 524
  - and interfaces (*see* Modularization and interfaces)
- Modularization and interfaces
- formula student case
    - automatic naming algorithm, 436
    - generated interface graphs model, shifting system, 437, 438
    - interface graphs model, shifting system, 437, 438
  - MSI, 436
  - power system, 436–437
  - SAE shifting system, 434, 435
  - modularization and interfaces, 434
  - workflow and function modeling
    - automatically generated interface, 432, 433
    - automatic naming algorithm, 432, 433
    - business process modeling, 432, 433
    - DSM, 434
    - flowchart-type description, 432
    - functional system decomposition, 432
    - simplified workflow model, 431, 432
- Modular launch planning
- digital cameras, 621
  - module drivers, 620
  - product launch plan, 620–622
  - Sony Handycam range, 620, 621

- Modular product families, integrated
  - PKT-approach
  - control devices, industrial trucks (*see* Control devices, industrial trucks)
  - description, 250
  - gas inlet valves (*see* Gas inlet valves)
  - herbicide spraying systems
    - DSM, 248, 249
    - MANKAR-Roll, 247–248
  - method units (*see* Method units, integrated PKT-approach)
  - MIG, 251–252
  - MIM, herbicide spraying systems, 248–250
  - modern market situations, 245–246
  - modular lightweight design, 266
  - reduction, internal variety
    - DSM, 247
    - product modularization, 247
    - product platform, 247
    - product variants, 247
    - variety-oriented product design, 246–247
  - serial and parallel applications, 267
  - water measurement devices (*see* Water measurement devices)
- Modular product programs
  - CoC, 258
  - market-driven factors, 257
  - product structure strategies and commonality, 257–258
- Module
  - battery cells, 619
  - boundary definition
    - clustered DSM, UGV family, 335, 336
    - coupling and similarity, 334
    - definition, 334, 335
    - function deployment, 334
    - unclustered DSM, UGV platform, 334–335
  - generation, 617
  - interfaces, 548–549
  - liquid separator, 620
  - logic, architecture, 619, 620
  - MIM (*see* Module indication matrix (MIM))
  - PMM documents, 619, 620
  - technical solutions, 617
  - transparent/nontransparent versions, 620
- Module commonalization
  - and modular architecture, 686
  - optimal design (*see* Global product family design)
- Module Drivers
  - application, 99–100
  - battery packs, 100
  - building air conditioning system, 113
  - cell phone, 117
  - common unit and carry over addresses, 99
  - complications, 104
  - construction equipment accessory, 114
  - customer intimacy companies, 102
  - different specification, 100
  - hard drives, 101
  - information objects, 99, 101
  - laptop and e-readers/tablets, 100
  - laptop microprocessors, 101
  - medical industry, 98–99
  - and MIM, 103–104
  - modular function deployment, 96
  - motherboards, 101
  - operational excellence companies, 102
  - PALMA database, 103
  - product leadership companies, 102, 104–105
  - product life cycle stream, 96, 97
  - profile, 103
  - project team, 104
  - recycling, 102
  - riding-machine platform, 108, 109
  - service and maintenance, 99
  - styling and different specification, 99
  - technical evolution and planned design change, 99
  - value disciplines, 102–103
  - vehicle industries, 99
  - “Voice of Customer”, 96–97
  - “Voice of Engineering”, 97
  - “Voice of Manufacturing”, 98
  - “Voice of Supply Chain”, 98
  - webcams, 101
- Module indication matrix (MIM)
  - battery voltage, 616
  - company-specific application, 616
  - DC motor, 616
  - description, 610
  - herbicide spraying systems, 248–250
  - iteration, 103, 111
  - module drivers (*see* Module drivers)
  - PALMA application, 103, 104
  - and PMM, 606
  - product leadership, 610
  - replaceable battery, 617
  - riding-machine platform, 107
  - styling handle and escutcheon, 616–617
- Module Interface Graph (MIG)
  - 3D CAD data, 251–252
  - product variety, 253
  - and TEV, 256, 257



Module of activities (MoAs)  
 definition, 523  
 identification, 514  
 types, modularity, 510  
 Module Process Chart (MPC), 255, 256, 262  
 Module Strength Indicator (MSI), 436  
 MOGA. *See* Multi-objective genetic algorithms (MOGA)  
 MOPSO. *See* Multi-objective particle swarm optimization (MOPSO)  
 MPC. *See* Module Process Chart (MPC)  
 MPED. *See* Market-driven product family design (MPED)  
 MSI. *See* Module Strength Indicator (MSI)  
 Multidisciplinary design optimization (MDO), 29, 395  
 Multidisciplinary product development.  
*See* Architecture-centric design approach  
 Multidiscipline, PFD. *See* Product family design (PFD)  
 Multi-domain, 9, 88  
 Multi-objective genetic algorithms (MOGA)  
 chromosome representation, 276  
 commonality objective function, 279–280  
 consistency constraints, 277  
 crossover operators, 278  
 decomposition and parallelization  
 chromosome representations, 280, 281  
 commonality value, 282  
 crossover and mutation, 282  
 description, 282–283  
 fitness calculation, 280, 282  
 initialization, 280  
 iteration and termination, 282  
 replacement, 282  
 mutation operators, 278–279  
 non-dominated sorting GA (NSGA-II)  
 code, 275  
 Multi-objective optimization, 394, 407–408, 410–411, 739  
 Multi-objective particle swarm optimization (MOPSO), 370–371  
 Multi-platform design, 368, 370–377, 386  
 Mutation operators, 278–279

## N

National Security Cutter (NSC), 626, 627, 632, 634, 638, 640, 641  
 Naval architecture, 627, 628, 641  
 Nonstandard feature (NF), 159–160  
 The North American Industry Classification System (NAICS), 106

## O

Object-oriented design  
 abstraction, 550  
 aggregation, 553–554  
 application frameworks, 556–557  
 associations, 552–553  
 composition, 554–555  
 encapsulation, 550  
 inheritance, 550–551  
 message-sending, 551–552  
 object model, 550  
 polymorphism, 551  
 Obsolete feature (OF), 160–161  
 OEM. *See* Original equipment manufacturer (OEM)  
 OF. *See* Obsolete feature (OF)  
 Offshore Patrol Cutter (OPC), 626, 629, 632, 638–641  
 One-piece flow, 594  
 One-step approaches  
 CP<sup>3</sup> model  
 description, 311  
 optimization algorithm, 315–316  
 PSMF, 312–315  
 SIO, product family design  
 conventional mapping, 308  
 description, 307–308  
 design variables selection, 308  
 implementation, VSMF, 310–311  
 VSMF, 308–309  
 OPC. *See* Offshore Patrol Cutter (OPC)  
 Optimal commonality decisions  
 characteristics, Pareto front designs, 641, 642  
 characteristics, US Coast Guard's vessels, 626, 627  
 correlation inference, 628  
 design variables, 628  
 development and application, 626  
 discrete pareto front, 632–633  
 evolutionary optimization (*see* Evolutionary optimization)  
 fuzzy utility, 628  
 integer vector, 639  
 mission effectiveness/cost objectives, 629, 630  
 multi-objective fuzzy optimization, 628  
 natural commonality, 639, 640  
 net fleet savings objective, 630–631  
 NSC<sub>15</sub> and OPC<sub>15</sub>, 641  
 NSC mission requirements, 638–639  
 performance/cost, 640  
 product family design, 625  
 progression, evolutionary solution, 639

- ship design synthesis model (*see* Ship design synthesis model)
    - test application, 626
    - three-objective method
      - analysis, 640
      - Pareto surface, 641, 642
    - two-objective method
      - evolutionary algorithm, 640
      - optimization, 639
  - Optimal design
    - multi-objective optimization, 407–408
    - PCA, clustering pareto solutions, 408
  - Optimization
    - evolutionary (*see* Evolutionary optimization)
    - idiosyncrasies, 450
    - interactive visualization methods (*see* Interactive visualization methods)
    - multi-objective, 407–408
    - product family design, 450
    - and UTC product design space, 454–455
  - Optimization, platform and non-platform ECs
    - calculated sensitivity indices, 357, 359
    - constraints, 354–355
    - degree of customer satisfaction, 362
    - industrial pincers
      - calculated coefficients, 357, 359
      - engineering measures and benchmarking information, 356, 358
      - HoQ, 357
      - systemic image, 355–356
    - literature, 357
    - maximal OCS and OCS loss, 357, 360
    - objective function, 353–354
    - optimal values, product profiles, 357, 359
    - problem definition, 352–353
    - proposed approach, 360–361
  - Original equipment manufacturer (OEM)
    - automotive business, 125
    - consumer market, 122
    - suppliers, 121
- P**
- PALMA. *See* Product architecture lifecycle management (PALMA)
  - Parallelization, 246, 280–283, 291, 296
  - Pareto band
    - approach, 466
    - design bandwidth, 466
    - family and individual product, 466
  - Pareto front, 285–287, 632–633
  - Pareto set, 286, 370
  - Particle separator. *See* Aerodynamic particle separators
  - Particle Swarm Optimization (PSO), 311, 314, 315, 317
  - Part sharing, 63, 477
  - PBCM. *See* Process-based cost modeling (PBCM)
  - PCI. *See* Product line commonality index (PCI)
  - PDM. *See* Product data management (PDM)
  - Penalty function, 297, 298, 307, 310, 370, 518, 520, 527, 528, 635
  - Petri nets (PN). *See also* Timed colored petri net (TCPN) model
    - description, 507
    - handling process varieties, 518
    - seven-tuple, 514
  - PFD. *See* Product family design (PFD)
  - PFEG. *See* Product family evaluation graph approach (PFEG)
  - Platform
    - industrial contexts, 121–123
    - literature, 475–476
    - product development, 504, 530
    - requirement analysis stage, 504
  - Platform based product development
    - commonality vs. variety, 11–14
    - configuration and optimization (*see* Platform configuration and optimization)
    - description, 11
    - metrics (*see* Metrics)
    - powerful tools, 27
    - support systems and techniques, 27, 28
    - terminology, 27
  - Platform configuration and optimization
    - commonality, and modularity, 18–20
    - configuration and portfolio optimization problems, 14, 18
    - development, 14
    - families and platforms, classes, 18, 21
    - GBOM, 20
    - and non-platform parameters, 20
    - selection and design, 14, 17–18
  - Platform design
    - identifying and isolating algorithm, 209–211
    - lawn and landscape blower family (*see* Lawn and landscape blower family, PPM)
    - manufacturing improvement
      - functional efficiency, 211, 213
      - Lucas method, 211, 213
    - manufacturing tooling design, 213–214

- Platform design (*cont.*)
  - modeling, 211
  - nonphysical component platform, 214
  - optimization, 211, 212
  - performance, product family, 208
- Platform evaluation
  - negative valued targets, 375
  - optimization, product instances, 374, 375
  - positive valued targets, 375
  - stages, 374, 375
  - threshold value, 376
  - universal electric motors
    - optimization, 379, 382
    - performance, products, 382
    - products leveraged, 382, 383
- Platform investment, 24, 60, 62
- Platform leveraging, 8, 14, 327, 376, 384
- Platform modeling and configuration (PMC)
  - CCM software, 140
  - configurable components (CCs), 138
  - description, 137
  - and PDM systems, 138
- Platform optimization, 211
- Platform planning. *See* Product platform planning
- Platform relaxation
  - multi-platform design
    - evaluation stage, 374–376
    - mathematical model, 371
    - platform commonality variables, 371
    - relaxation stage, 376–377
    - single platform stage, 372–374
    - three-stage design process, 369, 371
  - stage, 376–377
  - universal electric motors
    - combined results, platforms 1 and 2, 384, 388
    - family, 384, 385
    - optimum design variables and performances, 384, 386
    - product evaluation, platform 2, 384, 387
    - strategy, case study, 384, 389
- Platform selection
  - and design, 17, 22
  - joint product (*see* Joint product platform selection and family design)
- Platform strategy
  - advantages, 49–50
  - automotive model, 49
  - Black and Decker's electric hand tools, 49
  - commonality (*see* Commonality)
  - divergence (*see* Divergence)
  - examination, 50
  - Joint Strike Fighter program, 49
  - low and high forward planning, 66, 67
  - market segmentation, 190–192
  - MQB platform, 49
  - product cost, 184–185
  - trade-offs
    - architectural parameters, 50
    - internal, 52–53
    - market, 51–52
  - water valve, 66
- Platform valuation
  - case study (*see* Case study, platform valuation)
  - company's profit model, 184–185
  - design quality, 187–188
  - financial model, 185–187
  - market-based design approaches, 180–182
  - modular product architecture, 196 and PFD, 181
  - product family architecture, 183
  - real options analysis, 180
  - sharing and reusing assets, 179–180
  - strategy cost, 184–185
- PLM. *See* Product life cycle management (PLM)
- PMC. *See* Platform modeling and configuration (PMC)
- PMM. *See* Product management map (PMM)
- Postponement
  - mass customization technique, 600–601
  - product architecture, 601
- PPCEM. *See* Product Platform Concept Exploration Method (PPCEM)
- PPCTM. *See* Product platform constructal theory method (PPCTM)
- Primary components, 76–77, 79
- Principal component analysis (PCA), 396
- Proactive platform design method using modularity (PPM)
  - companies, 201–202
  - design affordance, 204
  - function strategy, 208
  - lawn and landscape blower family (*see* Lawn and landscape blower family, PPM)
  - Lucas method, 218
  - market research, 204–207
  - modular architecture, 203
  - modularity construction, 214–218
  - platform design (*see* Platform design)
  - product engineering design process, 202
  - product family, 203
  - product family planning, 207–208

- product platform, 202, 203
- product strategy, 202
- top-down approach, 203
- Process-based cost modeling (PBCM), 481–482
- Process platform, 3, 504, 781, 783, 786
- Process reuse, 786
- Product
  - and process qualities, 534
  - and production system platforms
    - activities, 130
    - bandwidth, 129
    - configurable component, 134–137
    - decomposition, 133
    - definition, 129
    - designs, 131–132
    - different design projects, 131
    - elaborations and encapsulations, 133, 134
    - extensive properties, 130
    - functional and nonfunctional requirements, 132
    - function-means (F-M) tree, 132, 133
    - models, 131
    - platform, 129, 130
    - technical system characteristics, 131
    - theory of domains (ToD), 131
    - theory of technical systems (TTS), 131
- Product architecture lifecycle management (PALMA), 103
- Product data management (PDM), 137, 138, 140, 779
- Product design. *See* Product family design (PFD)
- Product development
  - cost, 62, 72, 181, 481
  - strategy and tactics, 92–93
- Product evolution, 739
- Product families
  - building, low cost
    - low-cost family variation, 592
    - mass production methodologies, 591, 592
  - commonality metrics (*see* Commonality metrics)
  - designing (*see* Designing product families)
  - design processes (*see* Design process)
  - flexible building (*see* Flexible building, product families)
  - and mass customization, 601–602
  - prerequisites
    - standardization, 602–603
    - total cost quantification, 603
  - spontaneous supply chains (*see* Supply chains)
  - structuring and selling
    - CNC, 590
    - definition, 589
    - flexible operations, nonfamily products, 590
    - sales and marketing, 591
    - supply chain and operations, 589
    - variety building (*see* Variety)
- Product family
  - APFD model algorithm, 85–87
  - bill of materials, 478
  - commonality
    - metric calculation, 478
    - selection (*see* Commonality selection)
  - common components, 87
  - process planning, 83–84
  - and product platform design
    - capabilities, enterprises, 782
    - coordination demand and supply chains, 781
    - corporate-level product platform, 779–780
    - customer and marketing interaction, 778–779
    - development, 777
    - financial analysis, 782–784
    - internet, 781
    - manufacturing systems, 781
    - open architecture product and service, 784–787
    - trade-off, 782
- Product family architecture, 183
- Product family configuration, 10–11
- Product family design (PFD)
  - aerodynamic (*see* Aerodynamic particle separators)
  - back-end issues
    - description, 3
    - front-end and development issues, 4–6
    - reconfigurability, 27–29
    - redesign and design reuse strategies, 29–31
    - supply chain issues, 31–33
  - biological analogy
    - APFD model (*see* Associated product family design (APFD) model)
    - cladistics, 73
    - cladogram construction method, 83–84
    - coevolution, 73
    - commonality and modularity, 84
    - constituents, 73

- Product family design (PFD) (*cont.*)
- designers and engineers, 73
  - design feasibility and variant generation, 80–81
  - functional analysis, 76–77, 79
  - interspecies co-speciation cladograms, 74
  - liaison graphs, 83
  - market analysis, 75–76, 78
  - modularity analysis (*see* Modularity, PFD analysis)
  - physical assembly process, 84
  - structural analysis, 77, 80
  - “black box” simulations, 451–452
  - classification, 368
  - cluster and sensitivity analysis., 370
  - commonality selection, 451
  - common components and functions, 2
  - components/modules, 451
  - consideration, 34
  - data mining-driven product design, 148–149
  - demonstration and application, 87
  - description, 35–36
  - design and production complexity, 87
  - design space exploration, 451, 452
  - design strategies, 148
  - development enterprises, 367
  - and development issues (*see* Platform based product development)
  - distinctiveness, 451
  - and DSMs, 87
  - engineering design optimization (*see* Engineering design)
  - front-end issues
    - description, 2
    - development and back-end issues, 2, 4–6
    - market-driven, 8–9
    - modeling product families and platforms, 9–10
    - platform and product family configuration, 10–11
    - product portfolio and product family positioning, 6–7
  - global platform development, 35
  - gradient-based optimization methods, 389
  - heterogeneity, 87
  - low-cost communication infrastructure, 147
  - management and engineering aspects, 33
  - manufacturing and resource constraint, 35
  - mass customization, 2
  - “master assembly process plan”, 88
  - minimal loss, performance, 386
  - MOPSO, 370–371
  - multi-platform design, 386
  - optimization approaches, 370
  - optimization methods, 368
  - Pareto-optimal solution, 370–371
  - performance, 452
  - performance and commonality, 368
  - platform divergence, 33–34
  - platform planning
    - design variables, 296
    - efficiency, 296
    - genetic algorithms (GA), 296
    - method, 296
    - quantification, 296
  - platform relaxation, 371–377, 386
  - platform selection, 370
  - PPCEM, 370
  - product cost, 184
  - product features, 174–175
  - product lifecycle (*see* Product lifecycle)
  - product platform and sharing decisions, 149–151
  - relaxation formulation, 369
  - robust optimization approaches, 34–35
  - scale-based product family design
    - method, 390
  - sizing (*see* Product family sizing design)
  - stages, 2
  - standardization, 34
  - temporal market-driven preferences (*see* Temporal market-driven preferences)
  - top-down and bottom-up approaches, 3
  - translating customer, 149
  - universal electric motor, 377–386
  - UTC (*see* United Technologies Corporation (UTC))
  - valuation (*see* Platform valuation)
  - variety management, 71–73
- Product family evaluation graph approach (PFEG), 25
- Product family model
- module production, 399
  - module transportation, 399–400
  - production, 400–401
  - product sales, 402
  - product transportation, 401–402
- Product family planning, PPM
- generic product plan, 208, 209
  - life cycles, 207–208
  - optimal set, products, 207
- Product family positioning, 6–7, 345
- Product family sizing design
- cannibalization, 685
  - changing and demanding requirements, 684

- cost, flexibility, and market demands, 685–686
- knowledge-based methodologies, 686
- literature, 685
- market, 684
- modularization, 686–687
- multi-objective genetic algorithm, 687
- multiple scalable platforms, 687
- objectives, 684
- optimization, 687
- perceptions and expectations, 684
- product options, 684
- purchasing power, 684
- sizing (*see* Product family sizing design)
- successive quadratic programming (SQP), 687
- variants, 685
- Product family use cases. *See* Case study
- Product features. *See also* Product family design (PFD)
  - classification, 148–149, 157
  - consumer electronics, 157, 158
  - data mining model generation, 155
  - Mann–Kendal trend test, 157–158
  - nonstandard feature (NF), 159–160
  - obsolete feature (OF), 160–161
  - standard feature (SF), 158–159
  - time series product data, 152
- Product lifecycle
  - commonality decisions, 733
  - components, 708
  - constraints, flow balance (*see* Constraints and adaptability)
  - description, 708
  - end-of-life management (*see* End-of-life management)
  - objective function, 714–715
  - optimal take-back and recovery strategy, 714
  - optimization outcomes
    - cores, optimal amount, 725, 728
    - cost reduction, 732
    - end-of-life management, smartphone family, 725, 729–730
    - graphical representation, optimal solution, 725–726, 731
    - implications, 728
    - material input–output flow, 728, 731–732
    - objective value, 725, 727
    - refurbished Phone 4, 732
    - take-back and part procurement costs, 732
  - optimization problem, 714–715
  - smartphone family design (*see* Smartphone family design)
  - traditional family design approaches, 734
- Product life cycle management (PLM)
  - CAE and CAD systems, 138
  - Configurable Component Modeler (CCM), 137
  - design rationale models, 138
  - PMC and the PDM systems, 138
  - software tools, 138
- Product life phases. *See* Life phases modularization
- Product line commonality index (PCI), 11, 13
- Product line rationalization, 590
- Product management map (PMM)
  - dendrogram and color coded, 620
  - logic, architecture, 619, 620
  - mandatory interlinked matrices, 606, 607
  - MIM and Module Drivers, 95, 96
- Product Platform Concept Exploration Method (PPCEM), 370
- Product platform constructal theory method (PPCTM), 346
- Product platform development
  - architecting activity, 424–431
  - computational support
    - air-conditioning system, 428, 429
    - clustering pattern generation, 430
    - development, product architecture, 428, 429
    - ElectricHeatGeneration*, 431
    - FBS modeler, 427
    - geometric modeler, 428
    - modelers, 427
    - physical phenomena, 430, 431
    - structural and behavioral descriptions, 427
- Product platform planning
  - CP<sup>3</sup> method
    - continuous optimization frameworks, 311–316
    - electric motors family, 317–319
    - integration (*see* Comprehensive product platform planning (CP<sup>3</sup>))
  - creation, 296
  - development, 295
  - one-step continuous optimization frameworks (*see* One-step approaches)
  - optimization algorithms, 296–297
  - particle swarm optimization, 320
  - PFD methods, 296
  - quantification, 295–296, 319

- Product platform planning (*cont.*)
    - SIO method
      - continuous optimization frameworks, 307–311
      - electric motors family, 316–317
      - integration (*see* Selection-integrated optimization (SIO))
  - Product platforms
    - description, 533
    - DfHV (*see* Design for Human Variability (DfHV))
    - software engineering
      - principles (*see* Software engineering principles)
      - qualities (*see* Software qualities)
      - techniques (*see* Software engineering techniques)
  - Product program
    - modular product programs, 257–258
    - planning (*see* Product program planning)
  - Product program planning
    - CAP, 257, 260
    - MIGs, 257
    - procedure, 256
    - PSM, 256–257, 259
  - Product Property Matrix, 333
  - Product quality, 184, 187, 188, 207, 213, 344, 347, 451, 527, 534, 537
  - Product strategy, 202, 248–250, 779
  - Product structure strategies, 257, 258
  - Product take-back
    - buyback price and program, 711
    - cost, 711
    - end-of-life management, 708
    - manufacturers, 711
    - recovery processes, 710–711
  - Product variants
    - design process, 508–509
    - families, structuring and selling, 589
    - flexible operations, 590
    - GVS, 509–510
    - modularity and scaling, 753, 754
    - PPM (*see* Proactive platform design method using modularity (PPM))
  - Product variety
    - integrated PKT-Approach, 251–252
    - management, 71–73
  - Program families, 546–547
  - Program Structuring Model (PSM), 256–257, 259
  - Project module identification
    - DSM and DMM, 511
    - fuzzy clustering, 511–514
    - MoAs, 510
- Q**
- QFD-based optimization method, scalable product platform, 344
  - Quality function deployment (QFD)
    - and ANP (*see* Analytical network process (ANP))
    - application, 25
    - based optimization method
      - advantages, two-stage approach, 352–353
      - commonality indices, 348
      - description, 344–345
      - disadvantages, 363
      - EC (*see* Engineering characteristics (EC))
      - limitation, 364
      - planning process, 347
      - procedure, 349–350
      - product platform, 348–349
      - scalable product platform, 345–347
      - single-stage, 363
    - CNs and FRs, 25
    - components, 690
    - cordless handheld vacuum, 614
    - customer requirements, 8, 94
    - customers, 693–694
    - customer values, 606
    - HOQ features, 609
    - implementation
      - final designs, group 1, 700–701
      - fit designs, clusters, 701–702
      - modular architecture and literature review, 699–700
      - multi-instrument capability, 700
      - ranked customer requirements, 697, 699
    - market segment, 608
    - and MFD, 608, 609
    - model, 149
    - product properties, 608
    - voice of customer, 609
- R**
- Real options
    - decision-making method, 180
    - financial model, 196
    - platform design, 183
    - product development processes, 182
    - uses, 7
    - valuation, 180
  - Reconfigurable system design, 27
  - Refurbishment, 712, 713, 732
  - Requirement management (RM) system, 138

- Reuse
  - redesign and design, 32–31
  - software
    - component-based, 556
    - object-oriented application frameworks, 556–557
- Riding-machine platform
  - iterations, 107
  - module drivers
    - indication matrix, 107, 109
    - profiles/scoring, 107, 111
    - technical solutions, 108
    - value disciplines, 108, 110
  - product lines, 106–107
  - strategic objectives, 107
- RMSE. *See* Root-mean-square error (RMSE)
- RM system. *See* Requirement management (RM) system
- Robust design
  - materials and manufacture, 560
  - program implementation, 536
  - requirements specification, 536
  - statistical modeling and ergonomics/human factors, 560
- Root-mean-square error (RMSE)
  - parameters, regression, 565
  - regression, 575
  - standard deviation and mean, 568
- Run-time system composition, 671–673
- S**
- SA-CAD. *See* Systems Architecting CAD (SA-CAD)
- Savings from commonality, 638, 641
- Scalable product platform, QFD-based
  - optimization method
  - common components and modules, 345
  - design, 345
  - engineering characteristics, 345
  - integration, 347
  - MPED, 346
  - PPCTM, 346
  - product commonality, 346
  - research, 345
  - stages, 345–346
  - VBPDM, 346
- Scaling
  - commonality matrix blocks, 304
  - design configuration, 298
  - factors, 382
  - platform components, 777
  - platform parameter, 384
  - PPM (*see* Proactive platform design method using modularity (PPM))
    - variables, 169–170, 308, 311
- Secondary components, 77, 80
- Selection-integrated optimization (SIO)
  - application, electric motors family, 316–317
  - and CP<sup>3</sup> model, 318–319
  - one-step approaches (*see* One-step approaches)
  - product family design
    - conventional mapping, 308
    - description, 307–308
    - implementation, VSMF, 310–311
    - selection, design variables, 308
    - VSMF, 308–309
  - product planning
    - application, 297–298
    - optimization problem formulation, 299
    - penalty function formulation, 298
    - universal electric motor family, 298
    - VSMF scheme, 297
- Self-organizing map (SOM), 396
- Sensitivity analysis, 189, 194, 346, 351–352
- Sensitivity Index (SI), 349, 351, 359
- Service-based decomposition, 227
- Setup elimination, 593
- SF. *See* Standard feature (SF)
- Ship design synthesis model
  - CERs, 633
  - parametric models, 633
  - power plants, 634
  - variables and ranges, 633, 634
- SI. *See* Sensitivity Index (SI)
- Single platform design, 386
- Single platform stage
  - binary decision variable, 373
  - formulation, non-platform specified
    - optimization, 372, 373
  - goal programming model, 374
  - platform-specified formulation, 373
  - universal electric motors
    - family, 379, 380
    - optimization formulation, 379, 381
- Singular value modularity index (SMI), 24–25
- SIO. *See* Selection-integrated optimization (SIO)
- Sizing
  - architectural module, 338–339
  - bicycle saddle height (*see* Bicycle saddle height)



- Sizing (*cont.*)
  - specification
    - equal accommodation, 567
    - equal variability, 566–567
    - military gloves, 565
  - tool handle, 573–576
- Smartphone family design
  - decision making, 733
  - economies scale, recovery operations, 733
  - high-sharing, part composition of product variant, 722, 723
  - optimization outcomes (*see* Product lifecycle)
  - parameter setting
    - disassembly yield rates, 723–724
    - product take-back information, 722–723
    - recovery cost assumptions, 724–725
    - recovery revenue and demand assumptions, 724, 726
  - structure, 721–722
- SMI. *See* Singular value modularity index (SMI)
- Software–abstraction, 542, 649
- Software–design for change
  - abstract machine, 546
  - algorithms, 545
  - data representation, 545–546
  - peripheral devices, 546
  - program families, 546–547
  - social environment, 546
- Software engineering principles
  - abstraction, 542
  - anticipation, change, 543
  - generality, 543–544
  - incrementality, 544
  - methodologies and tools, 538, 539
  - modularity, 540–542
  - rigor and formality, 539–540
  - separation, concerns, 540
- Software engineering techniques
  - design for change
    - abstract machine, 546
    - algorithms, 545
    - data representation, 545–546
    - peripheral devices, 546
    - program families, 546–547
    - social environment, 546
  - designs
    - abstract data types, 549
    - abstract objects, 549
    - bottom-up design, 548
    - module interfaces, 548–549
    - top-down design, 547–548
  - object-oriented design (*see* Object-oriented design)
  - principles (*see* Software engineering principles)
  - Reuse (*see* Reuse)
- Software frameworks
  - family platform, 649–650
  - industry-standard enterprise architecture, 648–649
  - software infrastructure, 649
- Software–generality, 543–544
- Software–incrementality, 544
- Software-intensive product platforms
  - advanced feature, 647
  - architectural code generation, 661
  - architecture objects to design patterns mapping, 660
  - code reuse (*see* Code reuse, product platforms)
  - constraints and adaptability (*see* Constraints and adaptability)
  - degrees of freedom (DOF), 647–648
  - design pattern structure, 660–661
  - develop product platform use cases, 654–655
  - frameworks (*see* Software frameworks)
  - general architecture (*see* Software layered architecture)
  - high-level process, product family platform, 654
  - industrial machines (*see* Industrial machines, software platforms)
  - knowledge graph, cell phone, 656
  - knowledge to architecture mapping, 659–660
  - mission-critical systems, 648
  - previous product projects, 655
  - refined knowledge model, 656–657
  - survey relevant literature, 655
- Software layered architecture
  - advantages, 652
  - API, 653
  - family products, 652, 653
  - implementation, 653
  - maximizing code, 651–652
  - modern products, 650
  - online banking application, 651
  - pattern, 650, 651
  - standardized hardware, 652
  - structure, 650–651
  - technical risk, 653
- Software–modularity, 540–542

- Software qualities
    - correctness, 534–535
    - interoperability, 538
    - maintainability
      - evolvability, 537
      - repairability, 536
    - portability, 538
    - product and process, 534
    - reliability, 535
    - reusability, 537–538
    - robustness, 536
  - Software–rigor and formality, 539–540
  - Software–separation of concerns, 540
  - SOM. *See* Self-organizing map (SOM)
  - Standard feature (SF), 158–159
  - Standardization, prerequisites, 602–603
  - Standard parts
    - and fasteners, 595
    - mass-customized parts and modules, 601
    - and materials, 599–600
    - product development teams, 598
    - steady flow, 595
  - Strategic axes, 103
  - Strategy
    - business, 117
    - enterprise-driven, 148
    - function, 208, 761–763
    - and market analysis, 190–192
    - module drivers, 99–101
    - optimization, 634
    - platform (*see* Platform strategy)
    - platform strategy cost, 184–185
    - “strategic suppliers”, 114
    - and tactics, 92–93
  - Supply chain configuration
    - and integration, product family, 396–397
    - and module commonalization, 406, 416
    - product family design, 31, 32
  - Supply chain management, 31, 33, 272, 595, 787
  - Supply chains
    - breadtruck, 596
    - dock-to-line deliveries, 597
    - flexible operations, 595
    - kanban, 596–597
    - linear cutoff, 595–596
    - material cut-to-length/shape, 596
    - min/max resupply, 596
    - on-demand in-house, 597
    - steady flow, standard parts, 595
    - strategic stockpiles, 597
  - Survey
    - agricultural workers, 561
    - consumer/user, 206
    - creation and deployment, 691
    - descriptive statistics, 695
    - focus groups, 149
    - market, 350
    - software designers, 655
  - System architecture
    - and architectural descriptions, 439–440
    - designing platforms, 423
    - development activities, 440–441
    - ISO/IEC 42010 standard, 422
    - mapped functionality, 438, 439
    - model-based development, 437
    - model-based paradigm
      - architecture-centric development process, 442
      - domain-specific models, 442
      - domain-specific tool, 443–445
      - information exchange, 441, 442
      - MATLAB, 441
    - modules, 336
    - product architecture, 421
    - proposed modeling language, 437–438
  - System bandwidth, 124, 127, 129, 140
  - System family, 126, 129, 135, 137, 139, 143
  - System modeling, 137
  - Systems Architecting CAD (SA-CAD)
    - architecting activity, 427, 428
    - physical phenomena search and clustering pattern, 430
    - product architecture, 429
- T**
- TCCI. *See* Total constant commonality index (TCCI)
  - TCM. *See* Total commonality metric (TCM)
  - TCPN model. *See* Timed colored petri net (TCPN) model
  - TEC. *See* Turbine exhaust cases (TEC)
  - Technical solution
    - air conditioning system, 111
    - DPM, 609–610, 614–616
    - MFD, 94, 606–607
    - MIM, 96, 616
    - modularity, 107
    - PALMA, 103
  - Technology
    - development platform, 126
    - ERP and PDM, 779
    - laptop microprocessors, 101
    - “Network Protocol”, 658
    - platforms, 127–128
    - roadmap, 336–337
    - “Technology Push”, 97, 260, 620

- Technology (*cont.*)  
 TRL, 123–124  
 UTC (*see* United Technologies Corporation (UTC))
- Technology roadmap, 336–337
- Temporal market-driven preferences  
 aerodynamic, 168–169  
 engineering design optimization, 151  
 KDD (*see* Knowledge discovery in databases (KDD))  
 PFD, data mining predictive model, 172
- Text mining, 7, 25
- Timed colored petri net (TCPN) model  
 control  
   colored tokens, 516  
   enabling and firing rule, 515  
   predefined rules, 515  
   time properties, 516  
 decision support to design process  
   planning, 519–520  
 performance indicators, 518–519  
 seven-tuple representation, 514  
 variety handling mechanism, 516–518
- Time series  
 Entropy statistics, 158–160  
 product data, 152  
 “rapid design transfer strategy”, 52
- Total commonality metric (TCM), 12, 13
- Total constant commonality index (TCCI), 150
- Tree of External Variety (TEV)  
 MIGs, 256, 257  
 product variety, 252, 253  
 VAM, 261
- Trend mining, 28
- Turbine exhaust cases (TEC)  
 configurable components (CCs) images, 139, 140  
 definition, 139  
 fabricated structures, 139  
 jet aeroengines, 139  
 platform system software architecture, 140, 142  
 system design rationales, 139, 141  
 variants, 142
- U**
- UGVs. *See* Unmanned ground vehicles (UGVs)
- United States Coast Guard  
 OME, 629  
 test application, 626  
 vessel designs, 633
- United Technologies Corporation (UTC)  
 commonality selection, 455–456  
 objectives, 455  
 product design space and optimization  
   algorithm, 454–455  
 product family design, 453, 454  
 system-level simulation model, 453
- Universal electric motor  
 complexity, decomposition scheme  
   all-in-one approach, 288–289  
   function and variants, 290  
 decomposition, 285–286  
 description, 283  
 design requirements, 378  
 design variables and constraints, 283–284  
 generalization, 286–287  
 minimum performance loss, 379  
 objectives, 284  
 parallelization, 291  
 platform evaluation stage, 379–383  
 platform relaxation stage, 384–389  
 product family, requirements, 377, 378  
 product parameters, 377  
 scale-based product families, 378  
 single platform stage, 379–381
- Unmanned ground vehicles (UGVs)  
 clustered DSM, 335, 336  
 customer needs gathering, 328  
 Market segments, 326–327  
 unclustered DSM, 334, 335
- UTC. *See* United Technologies Corporation (UTC)
- V**
- Value disciplines  
 aligned Module Drivers, 102, 117  
 building air conditioning system, 113  
 cell phone, 117  
 construction equipment accessory, 115  
 customer intimacy companies, 102  
 dimensional space, 93  
 operational excellence companies, 102, 103  
 product leadership companies, 102  
 riding-machine platform, 108, 110  
 space and intersection, 104
- VAM. *See* Variety Allocation Model (VAM)
- Variable segregating mapping function (VSMF)  
 description, 308–309  
 one-step approaches (*see* One-step approaches)
- SIO (*see* Selection-integrated optimization (SIO))

- Variant design
    - data mining-driven product design, 148
    - design process modeling, 505
    - market requirements, 75
    - optimization approaches, 273
    - process structure
      - design process, 508–509
      - GVS, 509–510
    - stages, 504
  - Variant management, 71–73
  - Variation-based platform design method (VBPDM), 346
  - Variety. *See also* Commonality
    - mass customization, 601
    - postponement, 600–601
  - Variety Allocation Model (VAM)
    - CoC, 265
    - optimize product structure, 254
    - TEV, 261
  - VBPDM. *See* Variation-based platform design method (VBPDM)
  - Viper Case study
    - ANP, 694
    - cluster analysis
      - customer groups, 697, 698
      - group membership scenario, 696–697
      - outcomes, K-values, 695–696
      - Viper player groups, 697, 699
    - customer groups, 693, 697
    - data cleaning and interpretation, 691–692
    - methodology flow, 690
    - objectives, 689–690
    - product family sizing design, 684–687
    - QFD (*see* Quality function deployment (QFD))
    - survey, 694–695
    - survey creation and deployment, 691
    - validity and optimum number clusters, 692–693
    - wood violins and Viper electric violin, 688–689
  - V-model, 424, 425
  - VOC. *See* Voice of the customer (VOC)
  - Voice of the customer (VOC)
    - Module Drivers, 96–97
    - QFD, 94
- W**
- Water measurement devices
    - CAP, 260
    - flow measurement systems, 259, 260
    - MIGs, 260–261
    - product program planning, 259
    - PSM, 259
    - stakeholders, 261
  - White-box reuse profile, 555, 677, 678
  - 60 Whr 6-Cell Lithium-Ion Battery, 162, 164–165
  - Wingquist Laboratory (WQL) Chalmers
    - aeroengine subsystem supplier, 139
    - aerospace sub-supplier company, 125
    - development knowledge platform, 124, 125
    - product system development, 143
  - Workflow models, 432