Embedded Systems

Twan Basten
Roelof Hamberg
Frans Reckers
Jacques Verriet *Editors*

Model-Based Design of Adaptive Embedded Systems

Springer

Embedded Systems

Twan Basten • Roelof Hamberg • Frans Reckers
Jacques Verriet
Editors

Model-Based Design of Adaptive Embedded Systems

Springer

Editors
Twan Basten
Embedded Systems Institute
Eindhoven, The Netherlands

Roelof Hamberg
Embedded Systems Institute
Eindhoven, The Netherlands

Frans Reckers
Embedded Systems Institute
Eindhoven, The Netherlands

Jacques Verriet
Embedded Systems Institute
Eindhoven, The Netherlands

ISSN 2193-0155 ISSN 2193-0163 (electronic)
ISBN 978-1-4614-4820-4 ISBN 978-1-4614-4821-1 (eBook)
DOI 10.1007/978-1-4614-4821-1
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2013930020

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

This book marks more than a decade of joint research by the Embedded Systems Institute and Océ-Technologies. The cooperation is based on a strong vision on model-based product development in which modelling techniques are applied to capture the design and to synthesise products. The HappyFlow paper path model is a successful example that allows Océ to capture sheet timing knowledge and to optimise productivity of several products. HappyFlow was one of the products of the first cooperation project called BodeRC, the predecessor of Octopus.

Océ delivers professional document processing equipment and software all over the world. Our products are used by commercial printing companies who rely on the quality and service of Océ products. The challenge is to offer reliable printing to all these different companies. All our printers operate in different circumstances with respect to applications, media types, media sizes, print volumes, and job lengths. Ideally a printer will adapt to the specific circumstances and deliver the best quality for each individual customer. For example, if on a particular site the power grid cannot deliver the load required for full performance, it would be great if the printer would reduce speed in order to stay within the available capacity. This adaptability poses an enormous challenge due to the many dimensions of circumstances and the complexity of dynamic in-product trade-offs.

The Octopus project has taken up this challenge to handle the multi-disciplinary complexity. The octopus is an intelligent animal and adapts easily to his environment and with his skilful tentacles it can handle many tasks. So Octopus as a name for the adaptability project surely reflects the ambition to enable development of intelligent and adaptive products.

At the start of the Octopus project we decided to strive for adaptability in-the-small as well as in-the-large, marked by three different lines of attention. An excellent example of adaptability in-the-small is the predictable control of the speed of individual droplets jetted by a printhead, adaptive to the content of the currently and previously jetted bitmaps. The way to balance in-product between system aspects like energy usage and productivity marks the other end of the spectrum. Predicting and comparing resource usage of a printer data path for all its use cases formed the challenging third line of attention.

The results are compelling and are directly visible in the early stages of product development. We are happy that the Embedded Systems Institute brings our product specialists together with researchers from many universities. Together they explore new areas in system modelling and push the boundaries for model-based product development.

Venlo
August 2012

Paul Hilkens
Vice President Mechanical Development
Océ-Technologies B.V.

Preface

It is with great pleasure that I welcome you to the final book on the ESI project Octopus, the last project funded under the Dutch BSIK programme Embedded Systems. This project has been executed by ESI, Océ, Delft University of Technology, Eindhoven University of Technology, the University of Twente, and Radboud University Nijmegen. The project started in July 2007, ended June 2012, and encompasses an overall volume of 92 FTE.

As for all of ESI's large projects, Octopus has followed the by now well-known industry-as-laboratory paradigm, in which scientific research is performed in the context of an industrial case. For Octopus, the case has been defined in the context of the development of high-end adaptive professional printing systems, thereby developing and using a model-based approach. Three main lines of attention were addressed:

- Data path design: the development and use of analysis techniques in order to verify whether functional and performance requirements of the data processing in a high-end printer can be matched with the hardware infrastructure. Key issue here is the wish to do these evaluations early in the design process, that is, at a point in time when making changes is still reasonably affordable.
- Advanced control for inkjet printheads: the development of high-end frequency-robust techniques to facilitate high-speed printing at high quality, i.e. with minimised spatial deformations of the intended dot patterns.
- System-level reasoning and design: the development and application of multi-disciplinary design methods and tools, to allow design trade-offs across disciplines, e.g. hardware, software, and mechatronics, in order to make better use of all available design options while managing the increased complexity.

The Octopus project has been highly successful. Among the results we count the following highlights:

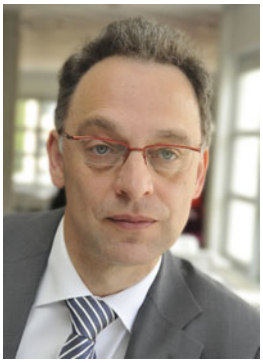
- An integrated data path evaluation environment, in which a variety of tools can be applied and combined, to validate required system behaviour, starting from a single model specification. The design model specification language hides the

underlying mathematical model analysis techniques, so that system designers (not necessarily analysis experts) can still apply these tools and do the appropriate analyses.
- An innovative control scheme for inkjet printheads that allows for higher-quality printing. This technique is being applied by Océ already in their newest printer design.
- An Océ patent application for printer controllers based on probabilistic models.
- A large set of overall system models, including the automatic verification of consistency between models from different disciplines, supported by tool prototypes.
- More than 100 scientific and professional publications, PhD theses and master's theses included.

All partners in the project are very satisfied with the results achieved in the Octopus project. Part of the activities in Octopus have found their way in a successor project Octo+, as well as in a number of national and international project initiatives.

Apart from the specific results within the project for the Octopus partners, more generally relevant results from this project will also find their way to the other projects ESI is executing, together with other industrial and academic partners.

I would like to thank all project participants for their commitment and contributions: as a team they have turned Octopus into a success! The support of Océ and the Dutch Ministry of Economic Affairs (now EL&I) through AgentschapNL, who together provided the funding, is gratefully acknowledged. We also thank Springer for their willingness to publish this book. With this book, we expect to share the important results achieved with a larger, world-wide audience, both in industry and academia.



Eindhoven
August 2012

Prof. dr. ir. Boudewijn Haverkort
Scientific Director and Chair
Embedded Systems Institute

Contents

Acronyms and Definitions

20-sim	Program for simulation of the behaviour of dynamic systems; derivation of Twente Sim
AD	Analogue to Digital
AI	Artificial Intelligence
AOP	Aspect Oriented Programming
API	Application Programming Interface
AUC	Area under (ROC) Curve
BAPO	Business Architecture Process Organisation
BodeRC	Industry-as-Laboratory project of ESI and Océ (2002–2006); acronym for Beyond the Ordinary: Design of Embedded Real-Time Control
BSIK	Public-private research programme funded by the Dutch Ministry of Economic Affairs, Agriculture, and Innovation; acronym for Besluit Subsidies Investeringen Kennisinfrastructuur
CLF	Control Lyapunov Function
CPN	Coloured Petri Nets
CPN Tools	Tool for editing, simulating, and analysing coloured Petri nets
CPU	Central Processing Unit
DA	Digital to Analogue
DDR	Double Data Rate (memory)
DMU	Decision Making Unit
DoD	Drop-on-Demand
DPML	Data Path Modelling Language
DSE	Design-Space Exploration
DSEIR	Design-Space Exploration Intermediate Representation
DSL	Domain-Specific Language
DSM	Domain-Specific Model
DSML	Domain-Specific Modelling Language
EM	Expectation Maximisation
ESI	Embedded Systems Institute
EU	Expected Utility

ESRA	Embedded Software Reference Architecture
FBS	Function-Behaviour-State
FCFS	First Come First Served
fifo	First In First Out
FIR	Finite Input Response
FPGA	Field-Programmable Gate Array
FTE	Full Time Equivalent
GPL	General-Purpose Language
GPU	Graphics Processing Unit
gsm	grams per square metre
GUI	Graphical User Interface
HDD	Hard Disk Drive
IDE	Integrated Development Environment
INCOSE	International Council on Systems Engineering
I/O	Input/Output
IP	Image Processing
JAR	Java Archive
KIEF	System architecting support tool; acronym for Knowledge Intensive Engineering Framework
LDS	Linear Dynamic System
LMI	Linear Matrix Inequality
MATLAB	Language and environment for numerical computation, visualisation, and programming; acronym for Matrix Laboratory
MIMO	Multi-Input Multi-Output
MO2	Multi-Objective Optimisation
MOO	Multi-Objective Optimisation
MPC	Model Predictive Control
MPU	Multi-Processor Unit
MRAC	Model Reference Adaptive Control
NP	Non-deterministic Polynomial time
Octo+	Industry-as-Laboratory project of ESI and Océ (2012-)
Octopus	Industry-as-Laboratory project of ESI and Océ (2007–2012)
	Tool set providing support to model, analyse, and select design alternatives in the early phases of product development
OSGi	Open Services Gateway Initiative
Paralyzer	Pareto analysis tool; acronym for Pareto Analyzer
PC	Personal Computer
PCB	Printed Circuit Board
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PFRS	Physical Feature-based Reasoning System
PI	Proportional Integral (control)
PID	Proportional Integral Derivative (control)
ppm	pages per minute
QPAS	Qualitative Process Abduction System

QPT	Qualitative Process Theory
R&D	Research and Development
RAM	Random Access Memory
RASDF	Resource-Aware Synchronous DataFlow
ResVis	Resource usage visualisation tool; acronym for Resource Visualisation
ROC	Receiver Operating Characteristics
RSI	Recursive System Identification
SA	Systems Architecting
SA-CAD	Systems architecting support tool; acronym for Systems Architecting Computer-Aided Design
SADF	Scenario-Aware DataFlow
SATA	Serial Advanced Technology Attachment
scheDL	Scheduling Domain-Specific Language
SDF	Synchronous DataFlow
SDF3	Tool for generation, analysis, and visualisation of synchronous dataflow graphs; acronym for Synchronous DataFlow For Free
SIDOPS+	Modelling language of the 20-sim tooling; acronym for Structured Interdisciplinary Description Of Physical Systems
Simulink	Environment for multi-domain simulation and model-based design
SISO	Single Input Single Output
SMC	Statistical Model Checking
SPI	Service Provider Interface
SPR	Strictly Positive Real
STD	State Transition Diagram
SVD	Singular Value Decomposition
TCO	Total Cost-of-Ownership
TCTL	Timed Computation Tree Logic
T-S	Takagi-Sugeno
TTF	Toner Transfer (belt)
UI	User Interface
UML	Unified Modelling Language
Uppaal	Model checker based on timed automata, named after its developers, the universities of Uppsala and Aalborg
USB	Universal Serial Bus
V-model	Graphical representation of the system development process; named after its resemblance to the letter V
XML	Extensible Markup Language
Y-chart	Graphical representation of the design-space exploration process; named after its resemblance to the letter Y

Chapter 1
Adaptivity in Professional Printing Systems

Jacques Verriet, Twan Basten, Roelof Hamberg, Frans Reckers, and Lou Somers

Abstract There is a constant pressure on developers of embedded systems to simultaneously increase system functionality and to decrease development costs. A viable way to obtain a better system performance with the same physical hardware is adaptivity: a system should be able to adapt itself to dynamically changing circumstances. The development of adaptive embedded systems has been the topic of the Octopus project, an industry-as-laboratory project of the Embedded Systems Institute, with the professional printer domain of Océ-Technologies B.V. as an industrial carrier. The project has resulted in techniques and tools for model-based development of adaptive embedded systems including component-level and system-level control strategies, system architecting tools, and automatic generation of system software. This introductory chapter presents the Octopus project and provides a reading guide for this book, which presents the results of the Octopus project.

J. Verriet (✉) • R. Hamberg • F. Reckers
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: jacques.verriet@esi.nl; roelof.hamberg@esi.nl; frans.reckers@esi.nl

T. Basten
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Electronic Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: a.a.basten@tue.nl

L. Somers
Océ-Technologies B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands
e-mail: lou.somers@oce.com

1.1 Introduction

The development of complex high-tech systems often endures tremendous pressure to get the job done. Frequently, the proposition "faster, better, cheaper" is put forward. Many manufacturers of high-tech systems are in a continuous rush to meet deadlines in offering the right product for the right cost at the right time. Innovation cycles must be kept short, mainly due to price erosion after having introduced a new product in the market. Accordingly, there is an imperative to simultaneously push new products and features onto the market while responding to evolving customer expectations regarding new applications and product improvements. Technology investments are most effective for a portfolio of products rather than for independent products. This entails the development of architectures and components that lend themselves to a series of products, often called a *product family* [5]. These architectures and components allow an adequate market response. However, because of legacy configurations and inflexible architectures, a product family approach has the risk of creating products that are rigid and brittle. The way to avoid this risk is to improve system *evolvability* [4], i.e. the ability to evolve in response to evolution of technology, competition, and customer expectations. Systems should be easily modifiable to take advance of new hardware and software technologies and to respond quickly to changing customer expectations.

Market and technology dynamics pose yet other challenges. For example, the unpredictability of changes in a system's environment and the change in use of a product pose increased evolvability requirements. Current development approaches specify the functionality of a system at design-time. The system is developed to operate reliably within a well-specified range of conditions; e.g. in terms of environmental conditions such as humidity and temperature, product performance limitations, or energy consumption. Such approaches are not adequate to develop systems that adapt to environmental or usage changes.

A viable answer to these new challenges is *system adaptivity*, i.e. the ability of the system to adapt itself to changes in its environment, system usage, etc. An adaptivity approach maximises a system's life-time value for various stakeholders. This is achieved by an extension of the system's operational range.

The key issue in adaptivity is to keep the system in an operational mode that allows it to perform its function in an optimal manner. The adaptivity strategy must not lead to inappropriate or unpredictable system behaviour. The main research question therefore is how to detect in what manner to adapt the product behaviour in its operational space and how to put this adaptivity into effect. The challenges of creating adaptive systems are manifold.

- What are appropriate approaches to model and analyse the adaptable properties of the system in its environment?
- What and how to measure the system during operation?
- What are the guidelines, approaches, and constructs to develop adaptive systems?
- How to define an adaptivity strategy, based on the measurements, adaptivity possibilities, product environment, etc.?

- How to control software and hardware to adapt the system's behaviour according to the defined strategy?
- System adaptivity may lead to unpredictable emergent behaviour. How to guarantee correct system behaviour?

1.2 Professional Printing Systems

Professional printers are high-tech embedded systems that have to work in a wide variety of environmental circumstances: e.g. they have to deal with different temperatures, humidity levels, and outlet power levels. Printing technology takes a set of blank media and a digital file as input and produces a printed document as output. The printing process includes a variety of technologies including inkjet and dry toner. Over the past years, digital printing technology has become mature and the market for digitally printed documents has become very diverse. The market includes home printing with low-cost printers, office printing, and professional printing. The media on which is printed is diverse in terms of size, mass, thickness, roughness, coating, finishing, etc. Printer manufacturers need to have in-depth understanding of the characteristics of each digital printing market segment to position the right product at the right cost-of-ownership in the right segment of the market.

The professional printer market involves systems that produce, distribute, and manage documents. Documents may be printed in colour or black-and-white and in a variety of formats (i.e. small or wide). Customers are typically found in offices, education, industry, and the graphics industry. As such, the market of professional printers starts at the top of the low-cost office printers and ranges up to offset lithography printers (e.g. used for printing of newspapers). Some examples of professional printers are shown in Fig. 1.1.

For professional printer manufacturers, market reputation is typically based on *productivity* (effective number of pages per minute), *reliability* (always operational), *quality consistency* (constant high quality and colour consistency), *ease of use*, and a competitive *cost-of-ownership*. In this environment, they have to anticipate changing market requirements, changing market conditions of their clients, competitor developments, and emerging market opportunities. Any company that plays this competitive game successfully, in terms of investment, innovation lead-time, life-cycle costs (development, manufacturing, service), and cost-of-ownership of the product, will emerge as a winner. Trends that have emerged in the professional printing market over recent year include the following.

- *Print-on-demand*. Quick turnaround and responsiveness to customer printing needs. This is enabled by electronic file submission. In-line finishing extends the capabilities to print complete products in a single pass, when, where, and how the customer demands.

Fig. 1.1 Examples of professional printers. *Top left*: Océ ColorWave 650 wide-format colour printer. *Top Right*: Océ VarioPrint DP black-and-white production printer. *Bottom*: Canon ImagePress C7010VPS colour production printer (Pictures from http://www.oce.com/ and http://cap.canon.nl/)

- *Variable data printing*. Digital printing allows printed materials to include information elements (images, graphs, charts) amidst static regions (text) that needs different processing, e.g. by providing localised high-quality output.
- *Distribute and print*. Electronic distribution of files and printing close to the point of need reduces postal or manual transport, thereby reducing distribution costs by moving the point of production closer to the point of delivery.
- *Colour*. There is a continuous focus on value and attributes of printing. Colour is an added value that provides a clear differentiator with respect to black-and-white printing. Today, most research and industry exposure concerns full colour. However, it is the associated cost price that still keeps the black-and-white market dominant. It is expected that in the near future the turn-over point from black-and-white to colour will be reached. A similar turn-over point has already been passed in the home printing market many years ago.
- *Higher print quality expectations*. Printer manufacturers are challenged to produce printed documents with the appropriate printing characteristics at an acceptable price. The professional printing market has a high demand for print consistency and print quality. There are many external and internal parameters that influence consistency and quality (e.g. humidity, temperature, speed) and they have to be controlled. The end user demands need to be balanced with the possibilities and restrictions imposed by the printer technology and implementation. Certainly at the high end of the printing market, where digital printing starts to compete with offset printing, the quality standards set forth by the offset industry, provides a key challenge.

- *Media range*. In order to offer new and exciting applications and move the market perception away from a commodity service to a value-added process, an increasingly wide range of media must be made available. An increasing range of paper sizes, masses, colour, texture, and finishing is needed. Complex printing jobs may include several media stocks in one pass requiring a wide media range concurrently within a print job.

Many of these trends influence the architecture and design of professional printing systems, requiring architectures and designs that can cope with change. Change is required during design-time due to changing market requirements. Moreover, change is required at run-time; e.g. to cope with changes in the environment that influence the print quality and consistency or to cope with other media. Furthermore, gradual change is needed to cope with wear of system components.

1.3 The Octopus Project

To address the adaptivity challenges listed in Sect. 1.1, the Embedded Systems Institute started the Octopus project in July 2007. The Octopus project has been carried out according to the *industry-as-laboratory* philosophy [3]: to obtain results that can be usefully applied in an industrial setting, the project has been conducted in the context of a concrete industrial case. The Octopus project has been conducted within the professional printing domain, which was introduced in Sect. 1.2. Océ-Technologies B.V., manufacturer of professional printing systems, has acted as the *carrying industrial partner* of the Octopus project. The other partners in the Octopus project have been Delft University of Technology, Eindhoven University of Technology, Radboud University Nijmegen, and the University of Twente.

The Octopus project has taken up research into the development of adaptive embedded systems. Its research has addressed the following topics with a special attention for system adaptivity:

- Strategies for component-level and system-level control;
- Methods and tooling that support model-based systems architecting and design;
- Methods and tooling that allow model-driven design-space exploration; and
- Methods for model-driven software engineering and code generation.

To obtain industrially relevant research results, the Octopus project has paid attention to validation of its results in an industrial context. Océ's professional printing domain has been used to assess the industrial applicability of the methods and tools developed within the Octopus project.

1.4 Reading Guide

This book describes the results of the Octopus project. As its title *Model-Based Design of Adaptive Embedded Systems* suggests, the Octopus project has yielded methods and tools that support model-based development of adaptive embedded systems. As a system's control software generally provides the capability of a system to adapt itself, the results in the book are mainly applicable to the development of software-intensive embedded systems. Examples of such systems are the professional digital document printers, which we have chosen as the industrial carrier of the Octopus research. Although these systems have been selected as a means to validate the methods and tools developed within the Octopus project, the project's results are applicable in a much broader context.

This book has nine chapters, of which Chaps. 2 through 8, form the core of the book. Chapter 2 defines the notion of adaptivity and addresses its importance and the challenges coming from the development of adaptive embedded systems. This is done according to the *Business*, *Architecture*, *Processes*, *Organisation* (BAPO) principle [5]. Chapter 2 identifies the trade-offs that are to be made and techniques to be used in the development of adaptive systems. It provides many examples of these trade-offs and techniques both within and outside this book.

Techniques and tools developed in the Octopus project are described in Chaps. 3 through 8. Each of these chapters provides a survey of an important aspect in the development of adaptive embedded systems and introduces techniques to handle this aspect. These surveys are generally accessible, meaning that detailed (mathematical) proofs and treatments have been omitted. Each of the book chapters is self-contained; in other words, they are understandable independently from the other chapters. The focus of chapters is on validation in an industrial context. Therefore, the chapters follow the *high-level method*, which relates industrial problems and goals to academic research questions and the corresponding (industrial) validation of (academic) research results. This research methodology was developed within the BodeRC project [1], an earlier industry-as-laboratory project of the Embedded Systems Institute and Océ.

1.4.1 Control of Adaptive Embedded Systems

The results of the Octopus project can be grouped into two main categories. The first category matches the *Architecture* aspect of BAPO (as explained in Chap. 2). It addresses the *control of adaptive embedded systems.* Adaptive embedded systems have to be able to deal with a large variety of different inputs. For instance, professional printers need to handle many different media and even more different documents. In addition, they have to function within a wide range of environmental circumstances, e.g. temperature, humidity, and outlet power level. Typically, an embedded system has many parameters with which it can adapt its behaviour. The challenge of controlling the dynamic configuration of these parameters taking into account all physical constraints is addressed in Chaps. 3–5 of this book.

Chapter 3 considers the control of professional inkjet printers. In particular, it studies the control of these printers' inkjet printheads. To allow a consistent high print quality for any bitmap, the printheads' nozzles must be able to generate a constant-size ink drop at any moment. A printhead nozzle has to correct for different printing speeds, bitmaps to be printed, and vibrations from itself and its neighbouring nozzles. This challenge is addressed by model-based design of robust nozzle-actuation pulses that effectively optimise the nozzles' jetting behaviour without the need for additional measurement equipment.

Where Chap. 3 addresses component-level control techniques, Chap. 4 considers system-level control strategies. It describes and compares four adaptive control approaches: model-reference adaptive control, gain scheduling robust control, model predictive control, and Gaussian stochastic control. The complex heating behaviour of a professional toner printer is used to assess the industrial applicability of the different strategies. The chapter describes how such a printer can adapt itself to changes of the environment temperature, the media type, and available outlet power.

Control engineering, as discussed in Chaps. 3 and 4, is an important discipline in the development of (adaptive) embedded systems involving physical processes. The success of control engineering greatly depends on the availability of an accurate model describing these processes. Chapter 5 considers situations where such an accurate physical model is not available: it describes how one can apply probabilistic reasoning to deal with incomplete knowledge. The chapter presents methods to create a probabilistic model from measured data and use the created model for stochastic control. This is illustrated in the professional printing domain using a media-recognition case and an engine speed control case for a professional toner printer.

1.4.2 Architecting and Design of Adaptive Embedded Systems

The second category of Octopus project results involve methods and tools for *architecting and designing adaptive embedded systems*. This category corresponds to the *Process* aspect of BAPO (see Chap. 2) and is covered by Chaps. 6–8 of this book. The first of these chapters, i.e. Chap. 6, describes architecting and design results. The chapter describes a method and a corresponding tool to support the architectural phase of adaptable system development in a systematic and model-based manner. The method distinguishes static decomposition and dynamic behaviour of the system to assist architects in systematically generating and selecting subsystem decomposition candidates. Together, the method and tool define and verify candidate architectures of the target system, and manage the interactions across engineering disciplines. The chapter demonstrates the usage of the described architectural tooling for a production toner printer.

As the complexity of embedded systems increases, so does the number of possibilities to implement these systems in hardware and software. Chapter 7 considers systematic support during the early phases of design to cope with this complexity. It describes the Octopus tool set for model-driven design-space exploration. The tooling applies the Y-chart paradigm [2], which advocates a separation between application software functionality, hardware platform implementation choices, and the mapping of software functionality onto the hardware platform. The chapter shows how the Octopus tool set can be used to quickly specify and analyse design alternatives. As a result, the tool set enables fast design-space exploration for software-intensive embedded systems. Two running examples are used to demonstrate the developed tooling: a high-end colour printer with a general-purpose hardware platform and a black-and-white production printer with a custom-built hardware platform.

Adaptivity is a system property, which is generally enabled by its embedded control software. Adaptivity can be achieved by the close cooperation of different software components. Unfortunately, this has a negative effect on the complexity and maintainability of the embedded software. Chapter 8 proposes a systematic, model-driven approach for the development of adaptive embedded software that does not sacrifice the quality of the software system. The approach involves the usage of domain-specific languages in which a part of the system can be specified and from which the corresponding control software can be generated automatically. The model-driven software engineering approach is demonstrated using two cases in the professional printing domain. The first case involves a scheduling domain-specific language from which scheduling code can be generated. The second introduces a domain-specific language for multi-objective optimisation used to make on-line trade-offs between different system qualities given the available resources (e.g. available outlet power).

1.4.3 Conclusion

The last chapter of this book, i.e. Chap. 9, is a reflective chapter. This chapter summarises and evaluates the results of the Octopus project. Moreover, it evaluates the Octopus project itself. Chapter 9 considers the impact of the project on its partners and the lessons learned regarding the organisation of industry-as-laboratory projects.

After Chap. 9, there are two appendices. Appendix A lists the Octopus project partners and Appendix B provides an overview of the project's publications.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. Heemels, M., Muller, G. (eds.): Boderc: Model-Based Design of High-Tech Systems. Embedded Systems Institute, Eindhoven (2006)
2. Kienhuis, B., Deprettere, E., Vissers, K., van der Wolf, P.: An approach for quantitative analysis of application-specific dataflow architectures. In: Proceedings of the 1997 IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '97), Zurich, pp. 338–349 (1997)
3. Potts, C.: Software-engineering research revisited. IEEE Softw. **10**, 19–28 (1993)
4. van de Laar, P., Punter, T. (eds.): Views on Evolvability of Embedded Systems. Springer, Dordrecht (2011)
5. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer, Berlin (2007)

Chapter 2
Aspects of Adaptive Systems Engineering: A Professional Printing Case

Roelof Hamberg, René Waarsing, Twan Basten, Frans Reckers, Jacques Verriet, and Lou Somers

Abstract Adaptive systems engineering comprises two individual themes, adaptive systems and systems engineering, and their interaction. In the Octopus project, some challenges that arise from these themes have been addressed in the realm of professional printers. This chapter serves to place these challenges in a common context, which is done along the BAPO structuring principle (Business, Architecture, Process, Organisation). The main research challenges addressed in the project appear in the architecture and process parts of BAPO. For architecture, patterns for behaviour and self-reflection about behaviour are the most relevant elements in the context of adaptive systems. For the architecting process, support through models in a model-based paradigm brings advantages in specification, options exploration and analysis, and synthesis of adaptive systems.

2.1 Introduction

There are several definitions of systems engineering, one more complex than the other. We start from a concise definition as given by INCOSE (see e.g. [8]) that covers the essence: *An interdisciplinary approach and means to enable the*

R. Hamberg (✉) • F. Reckers • J. Verriet
Embedded systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: roelof.hamberg@esi.nl; frans.reckers@esi.nl; jacques.verriet@esi.nl

R. Waarsing • L. Somers
Océ-Technologies B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands
e-mail: rene.waarsing@oce.com; lou.somers@oce.com

T. Basten
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Electronic Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: a.a.basten@tue.nl

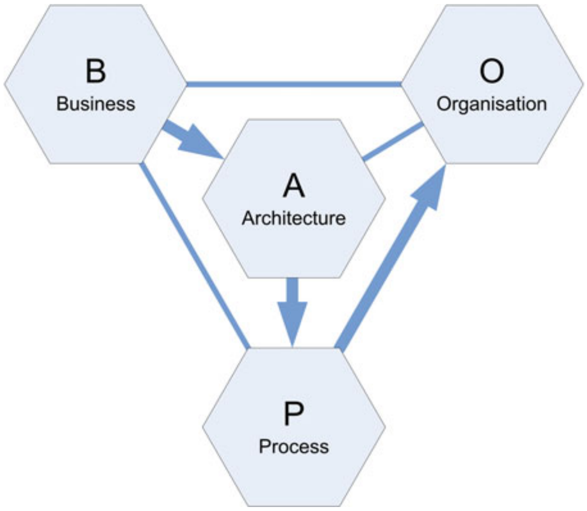


Fig. 2.1 Schematic representation of BAPO structuring with influence sequencing (BAPO concerns of software engineering; reproduced from [33] with kind permission from Springer Science and Business Media)

realisation of successful systems. It has a focus on the end result, i.e. a successful system, while *engineering* itself refers to the activities and tools needed to reach this goal. It is anticipated that multiple skills, applying different types of knowledge are needed to design and build such systems.

In the context of systems engineering, in general a structuring principle is necessary to coordinate the various engineering processes in an effective and efficient way. Such principle is called systems architecture, or reference architecture, or product family approach, dependent on the scope that one wants to address with the structure. All these variants use multi-view approaches, deal with many stakeholders, involve many system concerns, and embrace multi-technology solutions.

This book deals with aspects of system-level engineering of high-tech products that go beyond micro-electronics as regards the engineering disciplines required to develop them. In this chapter, these systems engineering topics are organised in a common framework. In order to do that, we also follow a structuring principle. The *Business*, *Architecture*, *Processes*, *Organisation* (BAPO) views [33], sketched in Fig. 2.1, are applied to provide a framework for the topics covered in this book. Especially the consequences of adaptivity of systems on the issues as brought forward through the BAPO views are indicated.

Special attention is devoted to adaptive systems in this book. The goal of adaptivity is to make a better system for its user, a system that operates well in different situations by adapting itself to them. In short, *an adaptive system is able to tune itself by responding to environmental changes*. For instance, a printer that prints at a lower speed whenever it detects that the electric power supply is lower than usual, is an adaptive printer. Note that words like *tune* and *change* hint at *dynamic* properties rather than *static* properties. We will see that *system behaviour* plays a central role in adaptive systems.

The typical challenges of systems engineering are related to keeping a predictive, comprehensive overview of all relevant system factors within an efficient process. The introduction of adaptivity of systems aggravates these challenges. Consider again the example of print speed adaptation triggered by variable mains: the speed control itself is somewhat more involved, but the largest impact arises from the interactions of print speed with the other printer parts and processes. Paper transport through the printer has to adapt, paper heating to prepare for toner transfer and fusion happens at changed rates, and even the image data transport channel has to be able to adapt its speed, to name just a few examples of affected processes. In short, keeping an overview is difficult for adaptive systems at all times, as it involves interrelated dynamic properties.

More formal and detailed definitions of adaptive systems exist (e.g. [6]), but the general idea is sufficiently clear: adaptive systems are more reactive to change by taking external information into account, thereby providing the user extended possibilities to operate the system. Along the BAPO structuring we further detail the notion of adaptivity. It is clear that all complicated systems are adaptive to a certain degree – here we focus on designing adaptive systems and their engineering processes in the case where a high degree of adaptivity is regarded as a first-level requirement.

In this book, based upon the systems engineering research topics addressed in the Octopus project, we highlight three types of challenges. The first set of challenges relates to the intrinsic complexity of adaptive systems: the concepts needed for adaptive systems are more difficult than for non-adaptive systems, and overall system consistency is harder to guarantee. An essential feature of adaptive systems is the presence of (control) feedback loops that exist in order to allow responses to changes at multiple (sub)system levels. This entails sensors, information aggregation, system state estimation, evaluation of what-if scenarios, codified knowledge about the system's working range, and actuators.

A second set of challenges relates to the process of developing adaptive systems. Keeping the increased complexity (as a result of adaptivity) under control, whilst working with multi-disciplinary teams to create different parts of the system, is a non-trivial problem. The research goal that we extracted from this challenge is to provide early feedback about the impact of design options and design decisions to systems engineers, as precise and accurate as economically feasible. A common starting point, also adopted by Octopus, is the (partial) virtualisation of means in the development process: physical prototypes of the system (components) are extended or replaced by simulations. Such models can provide an overview of dynamic properties in a much more flexible way. In this case, the software engineering process requires special attention, as adaptive system behaviour inherently leads to increased software complexity.

Thirdly, challenges arise from the need to be able to exchange components in adaptive systems at different moments in time. This relates to *system adaptability*, also called *evolvability* [32]: the degree in which a system itself can be changed. Adaptability leads to the full set of life-cycle considerations that deal with configuration management in product lines, component changes, and their subsequent calibration, changes of control, changes of system usage, and the validation of system behaviour. For the class of adaptive systems, adaptability is a complex property to manage, because of the dynamic character of the cooperation between the system components.

The three classes of challenges are introduced along a storyline that follows the views of BAPO. After the context setting through business reasoning in the next section, they are subsequently elaborated upon in the course of this chapter. We provide a general overview of the issues introduced by system adaptivity, although we obviously have worked from the specific context of professional printing and with the limited resources available in the Octopus project. The last two factors circumscribe the very broad field of adaptive systems engineering to arrive at the actual research topics of this book. In short, the reader will encounter a generic storyline that is illustrated by concrete examples, mostly from the domain of professional printing.

2.2 Business: Why Do We Need Adaptive Systems?

In the BAPO structuring principle, the Business view is where it all starts: the added value of a system for (the business of) its users and other stakeholders is the essential ingredient for its success. The Business should drive Architecture, Process, and Organisation, although typical rates of change may be slower in the latter views.

To put the added value of a system in context, consider the customer value model shown in Fig. 2.2 [18]. This simple model reflects the main factors that contribute to added value: increasing the benefits and/or decreasing the sacrifices and risks will lead to an enhanced added value.

Customer benefits represent tangible and intangible values of the system as perceived by the customer. Who is the customer? Obviously, there are multiple stakeholders, such as operators that use systems on a day-to-day basis, budget owners that decide on purchase of systems, service engineers that maintain systems, etc. All their requirements have to be taken into account. The customer as stakeholder is not a very clearly defined notion: the decision-making unit (DMU)[1] can be very different from the end user. This is represented in Fig. 2.2 by the timeline with different phases in which "customer benefits" involves different stakeholders.

Customer sacrifices are often expressed in total cost-of-ownership (TCO). This includes more than system cost only: the flexibility of the system to arrive at an appropriate application (from the perspective of the customer) as well as efficient serviceability are a few factors that play a role in TCO. Such sacrifices are regularly called *indirect costs*, the majority of which is related to time and effort that has to be spent in order to use the system. Here too, the different phases comprise stakeholders for whom the system requires different sacrifices.

[1]For a definition refer to Financial Times Lexicon (http://lexicon.ft.com/).

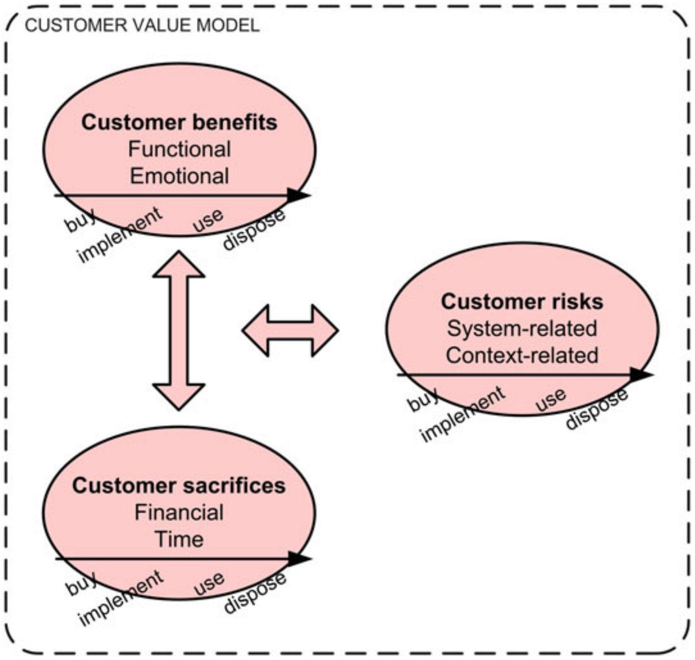


Fig. 2.2 Customer value model (adapted from [18]) illustrating the contributions to added value of benefits (positive), sacrifices (negative), and risks (negative). The balance between these contributions is dependent on the customer/product interaction phase as well, indicated by the timelines in the ellipses

The third category of factors for the concept of added value is indicated *customer risks*. Customers are inclined to accept a lower margin between benefits and sacrifices if they know the system will work, by itself and also in its context, i.e. the wider (super)system context it is used in. Risks relate to benefits and sacrifices: risks represent the uncertainties that are in the estimations/predictions of the actual benefits and sacrifices.

Following the customer value model, the business challenge is to enlarge the added value of systems. An R&D perspective on the customer model is illustrated in Fig. 2.3. From this perspective, the focus is what can be achieved in system development. System development can target at enlarging the operational working range of products (see also Sect. 2.3.2), i.e. better specifications and more functionality, or in lower investment in development, which translates into an important contributing factor of the customer's TCO, or a combination of both. A third attention point is the reduction of the customer risks, which can be addressed by predictability and application flexibility. The balance of these factors is dependent on target markets and the roles the products play in these markets.

Adaptive systems with working range extension naturally lead to systems that do not have a single optimal system performance under all conditions, but a variable,

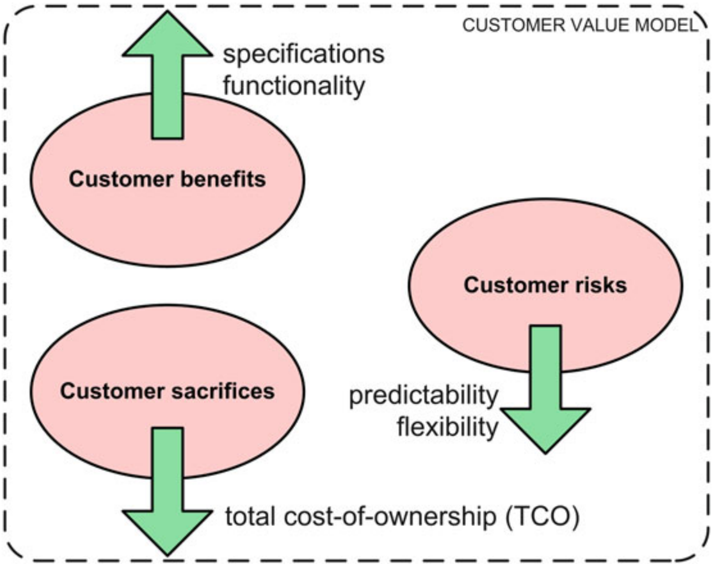


Fig. 2.3 Customer value model challenges from an R&D perspective

optimised performance within the scope of actual working conditions. The system optimisation will partly be customer-dependent in order to serve his needs best. It will also serve the DMU in the case of tenders in which the system is evaluated at a distance.

Let us consider some examples from the area of professional printing that illustrate increased added value by introducing adaptivity. The business context of professional printing involves customers who want to regulate their printing needs (either as a primary process for so-called commercial printers or as a secondary process for office-related services) in a professional way. Mostly this involves contractual agreements on printing systems, printing supplies, system upgrades, and service. High added-value propositions address efficient ways to generate substantial print volumes ($\gg 10^4$ prints/month), often reflected in very productive (>1 prints/s), robust, and user-friendly printers.

The first example is taken from a well-known fact in toner-based printing. The resulting print quality is good only if the media (paper sheet) temperature is in a certain range when fusing the toner image onto the media. With limited (media) heating capability this means that heavy media need more time, and thus, should be processed at a lower speed. The required adaptation of speed could be made continuously, but only if more actual information about the heater and the media is available, and preferably also information about their near-future status. This adaptation leads to improved customer experience (higher productivity), lower risk (media flexibility), and possibly lower cost (better utilisation of available resources like heating capacity, but also additional sensors and software in the system).

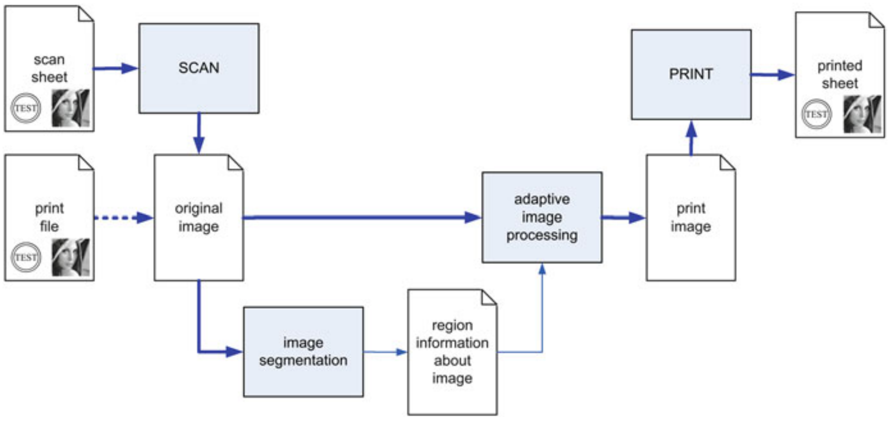


Fig. 2.4 Illustration of adaptive image processing in copiers and printers. The processing steps at the bottom add information to adaptively steer the image processing

A second example of adaptivity is the image processing of a document to achieve optimal print quality. The user choice between "text" and "image" originals for photocopiers is a familiar one. Because text images only contain sharp edges with high contrast, while "normal" images also contain low-contrast details, the optimal image processing requires different steps for these two cases. In an adaptive system, this processing can be steered automatically. For copiers, this requires scan-image segmentation (a complicated sensor in software to define text and image regions) in order to steer the image processing. Océ introduced a copier with this functionality in the 1990s. For printers, the digital information in the print file can be used as well. This adaptivity, illustrated in Fig. 2.4, results in an increased customer benefit (optimal print quality in all situations) and decreased customer sacrifice (less setup time before copying/printing needed).

A third example concerns adaptive ink jetting strategies. Draft, normal, and best printing modes on (inkjet) printers are familiar user settings. In fact, the printed image can be improved (for instance through multi-pass printing) at the cost of image reproduction speed and ink usage. In more recent printers, information about the status of the printhead as well as about the image content is utilised to improve the image quality without a loss in speed. Because of the redundancy that is present in printheads to put the right colour pixel at the right place, malfunctioning nozzles can be corrected for by other nozzles. Again, note the increased use of *self*-information to improve image quality at very limited cost for the user.

Looking at these examples, different features of adaptivity patterns can be discerned. Figure 2.5 attempts to sketch these patterns in a simple way. In general, the global functionality of the system is always enhanced by a dynamic property, either by providing better values for some aspect, or by providing more flexibility to the user to fulfil his actual needs. The more advanced examples of adaptivity build on the sensing of change: change of external parameters, either uncontrolled or

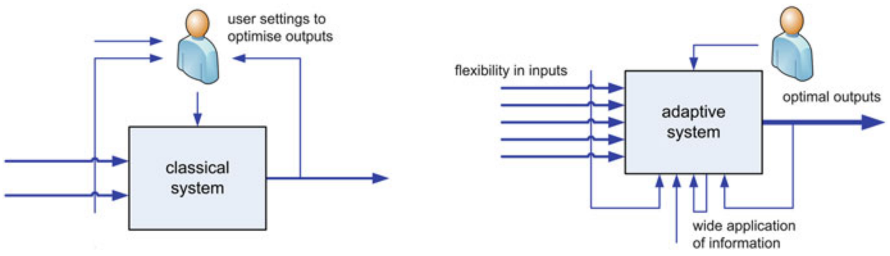


Fig. 2.5 Increased customer benefits in adaptive systems: a sketch of how it is realised. The adaptive system deals with a larger range of inputs, while the user has to provide less information. The system needs more sources of information about inputs, outputs, the environment, and itself

controlled by the end user. Sometimes the ability to sense changed input moves a choice for the user to an automated behaviour, just because it always yields a better end result [13]. Note the problem with *always*: the system-internal conclusion that is based on the sensory information should be of predictable high quality. Summarising, we state that adaptive systems engineering is used to increase the functionality of the system by allowing context-based optimisation, where the use of self-information greatly amplifies the benefits of doing so.

Why do we need adaptive systems? Most systems are already adaptive to some degree, and we have given an example-based business reasoning to increase the level of adaptivity even further. The bottom line from a business point-of-view is defined by products that provide a high functionality level while being more indulgent to a range of input conditions, either in an automatic way or through specific user choices. This indulgence lowers the TCO level of the system. In principle, two main solution directions[2] are observed: either the system is improved statically (components with better static specifications, most often increasing its cost), or the system is improved dynamically (components and system with changed behaviour, most often at the cost of system complexity). The second solution direction is business-wise more interesting, provided the development costs can be governed well. We have pursued our research mainly along this line: *how to put the full operational area of a system at the disposal of customers while their expenses in time and cost do not increase?*

We conclude that adaptivity does not come for free as adaptive systems are more complex: the real challenge in the development organisation is to keep control of the cost factor by applying suitable adaptivity patterns in the architecture and govern the life-cycle factors (development, assembly, installation, maintenance) through the right level of adaptability of the system-under-development. Only then, adaptive systems are the genuine solution to increase the level of system performance per unit

[2]Next to this, in the course of system life-cycles, regular cost engineering processes occur which decrease system cost as well, but generally while functionality is kept constant.

of cost. What we consider to be the most relevant elements of suitable architectures for adaptive systems is discussed in the next section; these elements are at the core of the research conducted in the Octopus project.

2.3 Architecture: What Techniques Do We Find in Adaptive Systems?

Architecture follows Business according to the BAPO structure. In the BAPO context, Architecture denotes the technical means to build the system. Here, in succession of the previous section in which the added value of adaptive systems was the main topic, specific technical concepts to build adaptive systems are discussed.

Adaptive systems tune themselves by reacting to environmental changes, as stated before. In other words, under different conditions the system can do something different in order to optimally serve its user, i.e. the system changes its behaviour. In this section, we first relate system behaviour to system design. Next, adaptivity can be applied to deal with variability; we explain what this means to us while introducing a number of terms that are used in this book. Ultimately, this leads to self-reflective models that enable the system to reason about itself, which is the topic of the last part of this section.

2.3.1 System Behaviour and System Design

Adaptive systems deal with variations in their context: provided inputs, required outputs, environmental conditions, and required user-system scenarios all contribute to the extensive set of variations. The goal of dealing with such variations was given before: we want to create better systems with better trade-offs, taking the dependence on context into account. The variations are dealt with through self-governed behaviour, i.e. the way the system behaviour is adapted by the system itself.

In order to understand what we mean with system behaviour, a small excursion is taken here. A system manifests itself through observed behaviour over time. Actually, functionality is always provided by behaviour of the system, and everything is geared towards controlling this behaviour in order to deliver the requested functionality. The structure of the system is chosen such that this can be done; the control algorithms are implemented with this in mind; dynamic properties of physical phenomena are applied such that functionality can be fulfilled. Ultimately, perceived system performance is provided through the system's behaviour.

How do designers establish this system behaviour? Two main factors can be identified here. First, any system, or in fact, any of its components, is subject to the laws-of-nature. This will cause dynamic behaviour when excitations occur.

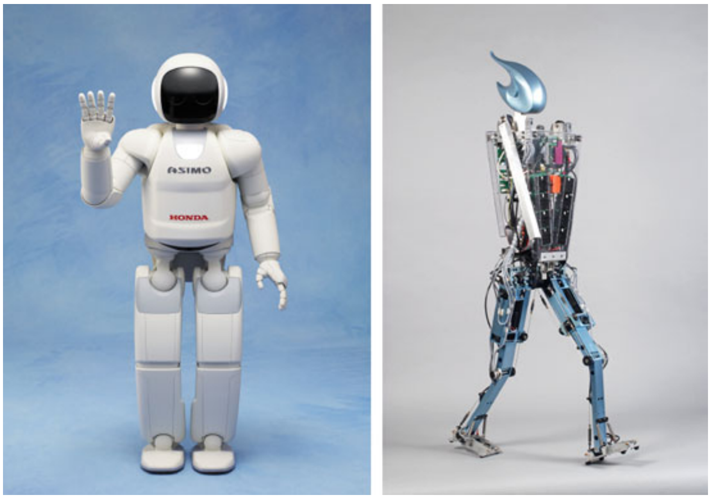


Fig. 2.6 The ASIMO robot of Honda [4] and the biped walking robot Flame of Delft University of Technology [14]

This type of behaviour we will call *natural behaviour*. For example, hot parts will eventually cool down, paper movement will slow down under the presence of friction, etc. These are the inevitable phenomena that will occur anyhow. Nevertheless, the limits to what system designers can achieve, are not entirely defined by these phenomena. That is to say, the second principal factor in system behaviour is the *imposed behaviour*. This behaviour is mainly effectuated through control software, although also in other engineering areas behaviour is imposed, such as in analogue feedback circuitry or passive switches [6, 21, 36].

Designers can control both contributing factors to influence system behaviour. The first factor is heavily influenced by the structural design of the system. A good example is given by the biped walking robots of Delft University of Technology [14]. Their geometry is designed in such a way that the walking movement is the result of physical laws without the interference of additionally imposed external control. The second factor is entirely versatile: almost anything is possible, although its practical limits caused by its necessary implementation (on a computing platform in the case of software) are not negligible. The ASIMO robot of Honda [4] is an example where the imposed behaviour through control software plays a much more dominant role. A picture of both robots is shown in Fig. 2.6. One has to realise that both factors can only be influenced up to the limits of the physical world, so much is clear.

An example that is relevant in the domain of professional inkjet printing has been one of the topics of research in the Octopus project. It concerns the design of the electrical pulses with which inkjet printhead nozzles are actuated in order to jet ink droplets. The parametrisation of these pulses with their control design on the one hand, and the physical dynamic properties of the ink channel on the other hand, together determine the behaviour of the inkjet printhead that immediately impacts the resulting print quality. The research into pulse control design has facilitated independence of jetting frequency and precise control of the droplet velocity, thereby resulting in improved print quality under variable conditions. This research is reported in Chap. 3.

2.3.2 Dealing with Variability: Keep Systems within Their Working Range

In the previous section, we have concluded that the design of system behaviour is essential for designing successful adaptive systems. This is a very abstract statement. In order to make it more tangible, system designers work with a limited set of physical quantities and properties that characterise the system and its behaviour. These characterisations come in three main types:

1. Attributes that are controllable, i.e. their values can be influenced in a direct way. In engineering these attributes are called *control parameters*, control variables, decision variables, etc.; in other contexts the term independent variables is used.
2. Properties that represent benefits for stakeholders, i.e. their targeted values are goals to be achieved by the system. In engineering these properties are called output parameters, *system qualities*, system properties, etc.; in other contexts the term dependent variables is used.
3. Contextual factors that have an influence on the system's operation. These factors cover constraints, job parameters, environment conditions, etc. We will refer to these factors as *context conditions*.

A property's type can depend on the context it is used in. Consider for example the fusing temperature for toner-based printing. Its value represents a target for the subsystem that prepares for printing by controlling heating power and other parameters. At the same time, however, it is a control parameter in the scope of the system, one of the factors that influence the targeted print quality.

A *working point* of a system is a vector of concrete values for control parameters that results in good operation of the system, which operation can in its turn be reflected in a set of appropriate values for system qualities. In control engineering the latter sets of values are called *setpoints*. Under different context conditions,

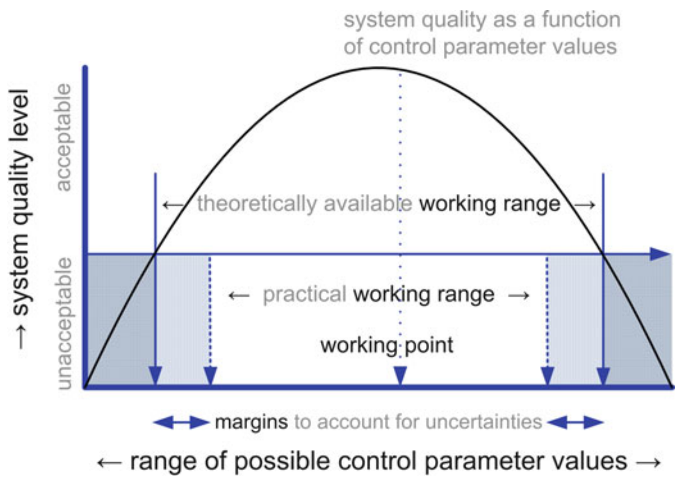

system quality as a function of control parameter values
→ system quality level
acceptable
unacceptable
← theoretically available working range →
← practical working range →
working point
margins to account for uncertainties
← range of possible control parameter values →

Fig. 2.7 A schematic representation of the concepts working point, working range, and margins, all for just one control parameter. Note that context conditions influence the quality curve and thereby influence the working range. These curves determine the margin – for the sake of clarity they are not drawn

different working points will be optimal. A regularly observed way-of-working in system design is that a set of working points is chosen to cope with different context conditions. Subsequently, at run-time the appropriate working point is selected, either by the user or by the system itself.

For small changes in the working point under stable conditions the system will also work well. The allowable changes to stay within certain limits of acceptable quality, define a *working range*, see Fig. 2.7. This working range is important for coping with the unknown: especially when context conditions are not known during operation, the designer has to take practical worst-case scenarios into account. In the literature, this relates to system-scenario-based design [12].

An example can serve to further illustrate the concepts. In the context of toner-based printing it is a fact that a toner image will be fused onto paper only if the temperature is within a certain range. The actual value of this temperature depends on control parameter values as well as on context conditions like environment temperature, heat transfer effectiveness, paper humidity, etc. If the latter conditions are not measured, a reasonable range has to be assumed for them, which in effect decreases the effective working range of the control parameters that regulate this temperature, because the target temperature range has decreased. It is clear that with less uncertainty about influential context conditions, the required *margin* within the working range is also smaller, leaving a larger working range to employ for system operation. Together, this defines how working point, working range, margin, and uncertainty are interrelated, a sketch of which is given in Fig. 2.7.

In virtually all practical cases multiple control parameters will be involved. This leads to more involved situations, sketched in Fig. 2.8, in which the values for

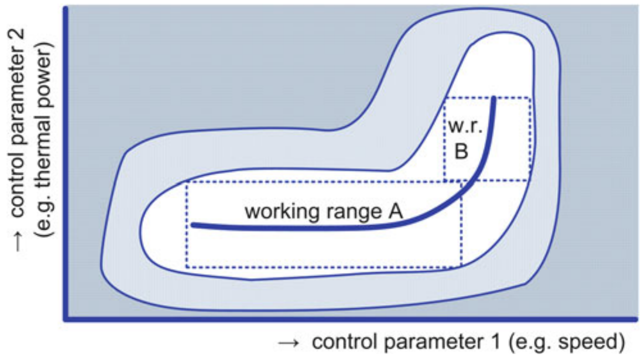


Fig. 2.8 A schematic representation of the working range and two possible practical working ranges for two independent control parameters. Note that the two working ranges reflect different trade-offs, while they are both within the available working range. The *bold line* indicates the set of optimal working points (cf. Fig. 2.9). Different context conditions will lead to different curves and working ranges

independent control parameters influence each other through the need for sufficient quality. As an example in the context of toner-based printing, consider print speed and heating power as two such control parameters. Only for large heating power a high print speed leads to acceptable print quality, because the fusing temperature should be sufficiently high and the time period to transfer heat to paper and toner is shorter in the case of a higher print speed.

The actual choice of values for multiple control parameters often reflects a trade-off between multiple system qualities. We referred to print quality in the previous paragraph, but at the same time the system qualities productivity and energy usage are influenced. Professional printing customers weigh these system qualities implicitly or explicitly with relative importance depending on their actual preference or context situation. In Figs. 2.7 and 2.8 we have combined the system qualities in one value; untwining the single value to multiple system qualities leads to so-called Pareto curves or Pareto fronts [25], illustrated in Fig. 2.9. Optimal values for system qualities (at the bold curve in Fig. 2.9) will coincide with control parameters in different working ranges (at the curve in the white area in Fig. 2.8).

The positive effect of having a larger working range upon increased predictability of behaviour can also be seen in an example taken from the data path in a printer, which takes care of transforming print files to printable images. The processing steps of a data path have variable durations, as they depend on the complexity of the image content and may be affected by resource contention. In order to guarantee sufficient throughput of the data path (to keep the system printing), the data path should be implemented

(continued)

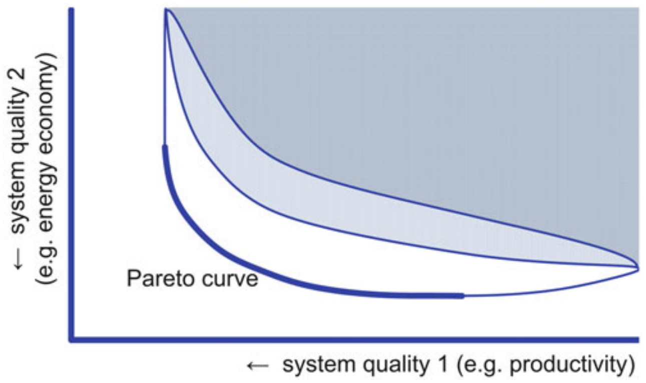


Fig. 2.9 A schematic representation of two system qualities and their attainable values through changing control parameters. Values closer to the origin are better in this representation; the *bold curve* denotes a Pareto front, consisting of a set of different optimal system qualities' values. The sketched areas are only indicative

(continued)
such that practical worst-case scenarios still work. More information about the actual complexity of images yields possibilities to use the available hardware resources such as a CPU and memory in a smarter way and to make better trade-offs between other system qualities such as print quality or productivity. Chapter 7 reports on data path models that facilitate studying the interrelationships between resources and system qualities.

Adaptive systems better utilise their working ranges through smart usage of their available resources, taking variations of different kinds explicitly into account. The latter is done by imposed behaviour that is enforced by control software. The level to which this is done may vary according to a classification given by Martín et al. [21] and displayed in Fig. 2.10, which discerns adaptation, anticipation, and rationality and intelligence.

Within the scope of *adaptation* [21], an accurate and precise estimation of the system's actual state through an increased number of sensors, but also aggregation of multiple data sources to more usable information is a necessary premise for successful adaptive behaviour. The explicit goal of being well informed (in the system, at run-time) is to improve the system's control and thereby, its behaviour. Next to adding more sensor-based sources of information, which introduces additional costs, another approach may be taken that strives to optimally use the already available information.

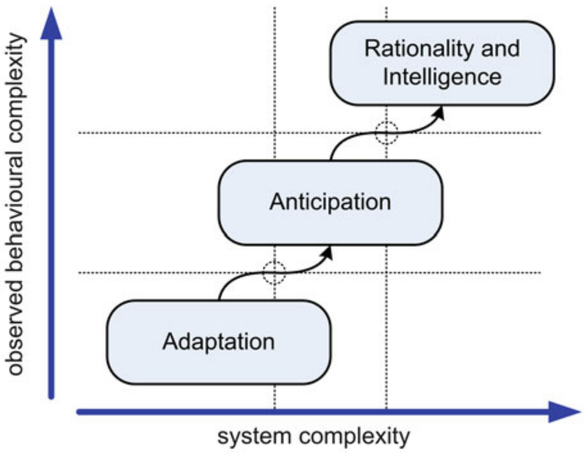


Fig. 2.10 The classification of adaptive systems as given by Martín et al. [21], following different degrees of observed behavioural complexity of systems

The necessary operational margins, based on practical worst-case reasoning, may be decreased by introducing reasoning with explicitly modelled uncertainties. The goal of this research direction in Octopus was to develop a method that results in tighter bounds on variable values, which are important for system control. Examples of such variables are temperature and paper mass. The first example uses these variables in a stochastic controller that applies a network of variables, their values being connected via conditional probabilities, to arrive at a better performance. The second example concerns a media classifier that works with inference on measured data. Chapter 5 deals with these topics.

The second level of adaptivity was coined *anticipation* by Martín et al. [21]. The basic idea is that one does not only have to react to external variations, but when applicable, known or expected future variations can also be taken into account for the current control actions to take. Classically, this is one of the many flavours of control theory, for example model predictive control (MPC) or model reference adaptive control (MRAC). In both methods, a model of the system is used to better align the control parameters with the required system qualities. Such methods are generally concerned with continuous variables controlling physical phenomena, and are highly optimised for the specific situation at hand.

A number of advanced control approaches are treated in Chap. 4, where they are compared to a traditional PI controller. Central to the findings in that

(continued)

(continued)
chapter is the fact that the process for which the behaviour has to be improved has to be studied. The salient characteristics will guide the designer in selecting the appropriate control approach that fits his design considerations. It is shown in a few subsystem cases for professional printers, in which the system can be described well in a model-based way, the system qualities print quality and productivity can be improved significantly.

The third level of adaptivity, called *rationality and intelligence*, involves behaviour that can be characterised as effective goal-oriented, not random, but unpredictable, according to Martín et al. [21]. This is outside the scope of this chapter and book. The first two levels of adaptivity both have to cope with on-line trade-offs, i.e. explicit decisions that reflect how different system aspects are weighted. This decision can be user-controlled or automatic.

2.3.3 *Self-Reflection: The Model of Your System in Your System*

As we have introduced adaptive systems in this chapter, they appear to be more knowledgeable about their environment and take sensible decisions that bring their behaviour towards fulfilling their goals. As Martín et al. already state in their definition [21], this means that there is a part of the adaptive system that relates the system goals to the system's adaptation.

This relation is not at all new. Every engineer will recognise such relations for his (sub)system. For "classical" systems, in many occasions the linking between goals and control actions is quite implicit. Engineers do the translation in their heads, while the result turns up in the form of a control scheme with selected variables, fixed parameter values, and implicit control objectives. For adaptive systems, it makes sense to explicitly reason about this relational part of the system. Amongst others, explicit models will constitute an essential ingredient to control the development of adaptive systems, topic of Sect. 2.4.

It is like putting part of the systems engineers in the system itself. The development of a system involves understanding the relationships of a range of properties and elements of the system, either through heuristics, measurements, hearsay, or explicit models. At least in development, systems engineers structure the knowledge for themselves to make it accessible for reasoning about the system behaviour. If the system has to do this reasoning itself at run-time, the knowledge structuring has to go – one way or another – into the system itself. Furthermore, in order to do incorporate this knowledge in a controllable way, it makes sense to consider it as a separate entity in the conceptual system architecture.

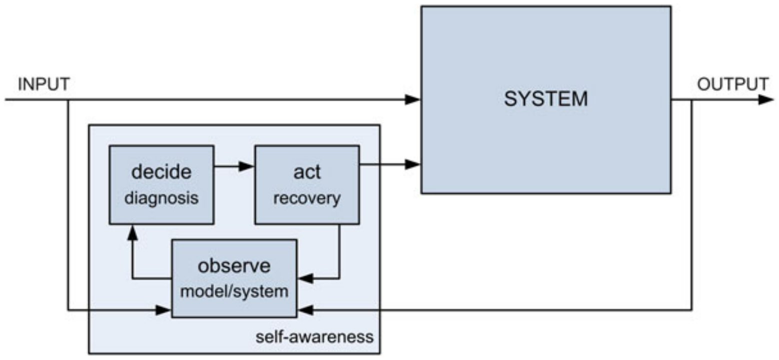


Fig. 2.11 Self-reflection in systems: combined reproduced scheme from [22] and [15]

What will this approach bring us? Does it improve on the business goals of the system? We think it will, because it introduces self-reflection possibilities and reasoning in an economical way. The imposed system behaviour, implemented through control software, can be better kept to-the-point, as the system model will take care of the translation to the real sensors and actuators. Moreover, what-if scenarios can be investigated by exploring the system model for future behaviour without really executing the control actions. These analyses give the opportunity to optimise over a range of possibilities without explicit codification of the control actions for the complete system's state space.

If self-reflection is such a good idea, why has it not been done before? Well, it has been done before, also at a regular basis, but as far as we can see mostly in a very implicit way. A notable exception to this are software systems based on belief-desire-intention (BDI) [26] software/agent technology (Brahms of Agent iSolutions [1], CogniTAO of CogniTEAM [5], JACK Intelligent Agents [35]), but they have a mainly mono-disciplinary nature. Another example is the SEEC framework for self-aware computing [15]. The Trader project [22] has reported on self-reflection as a means to improve the system reliability. The coinciding view of diverse sources is shown in Fig. 2.11. Here we come to a conforming conclusion: if the degree of adaptivity of a system is increasing, the separate notion of the system's model in the system is advantageous for system overview.

2.4 Process: How Do We Develop Adaptive Systems?

When Business and Architectural considerations have been made, the development Process has to be tuned to support these considerations. Viewing system development from the angle of the development process is a complex topic in itself: how are roles, responsibilities, and their relationships arranged? One may ask oneself why this topic is addressed at all: does it have anything special to do

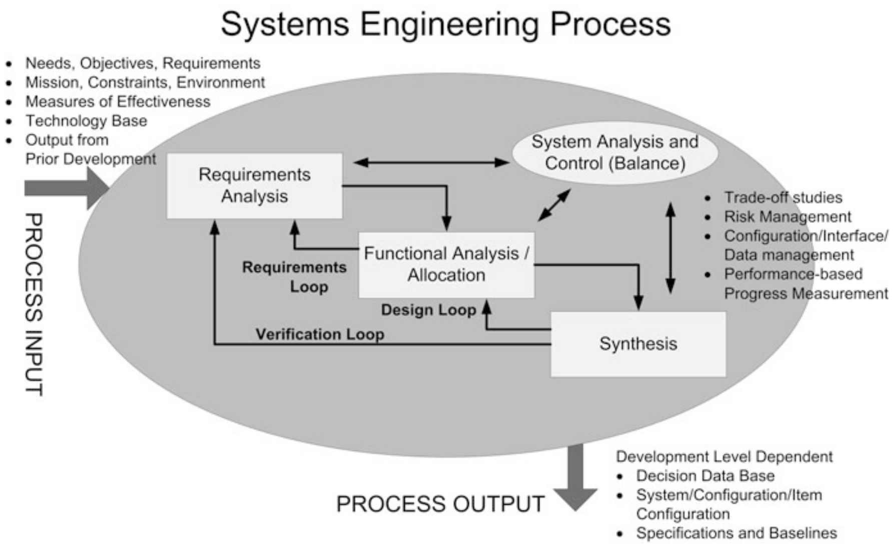

Systems Engineering Process
Needs, Objectives, Requirements
Mission, Constraints, Environment
Measures of Effectiveness
Technology Base
Output from Prior Development
PROCESS INPUT
Requirements Analysis
System Analysis and Control (Balance)
Functional Analysis / Allocation
Requirements Loop
Design Loop
Verification Loop
Synthesis
Trade-off studies
Risk Management
Configuration/Interface/ Data management
Performance-based Progress Measurement
PROCESS OUTPUT
Development Level Dependent
Decision Data Base
System/Configuration/Item Configuration
Specifications and Baselines

Fig. 2.12 The systems engineering process, a reproduced scheme from [30], which was later extended to the V-model, including integration and test phases

with adaptive systems? We answer this question affirmative and present a line of reasoning about the most challenging impacts that adaptive systems have in terms of their development process.

2.4.1 Challenges in the Systems Development Life-Cycle

It is common to divide the development process of complex systems in multiple phases, each with its own role. Many standards have been developed to regulate this process [3, 10, 20, 28, 30]. The most extensive standards cover the whole life-cycle of systems from initiation to disposal. Here we limit ourselves to the middle phases that are typically covered by the responsibilities of R&D organisations. These are shown in Fig. 2.12, which represents an early version of the DoD (Department of Defense) systems engineering process [30] that is also focused on the middle phases. In the scope of these middle phases, the early phases mainly have to clarify the system functionalities that suit the market and investigate what technologies can provide those functionalities. The later phases are focused on concrete realisation of the system, and responsibilities are often divided over the realisation of concrete, physical subsystems. In the current state-of-practice the structural division in concrete subsystems, also prominent through its physical presence, is dominant throughout all mentioned phases. This is a natural and successful strategy: when the interactions between subsystems are known and can be well described by static properties, such subdivision of responsibilities follows the well-known high-coherence/loose-coupling principle.

For adaptive systems, this subdivision still is important for managing the development process. On the other hand, due to the increased exploitation of interactions in adaptive systems, which is a result from striving to use the complete operational space (see Sect. 2.3.2), the divide-and-conquer approach on the basis of subsystems deteriorates. After all, not only the static nor stationary features are important, but in particular also the dynamic behaviour of the adaptive system. Especially the fact that in current practice the early phases (requirements analysis, architecting, functional analysis, design) are mainly document-centric processes, gives rise to additional complexity. For example, consider paper drying in an inkjet printer which may be mainly attributed to the dryer subsystem, but gets significant contributions from other subsystems, such as paper transport, cooler, and pre-conditioner. The relative contributions of these effects will vary over different (sub)system design options, over time, with different media, with different printed images. In an adaptive setting, this knowledge would be applied and hence, the dynamic control will have to extend across subsystems. If in the development process any of the involved subsystems change, this will have impact on the control. Summarising, whatever subsystems are defined, the coupling between them in adaptive systems is intensified with respect to non-adaptive systems, during system operation as well as during system development. Knowledge about such couplings should be actively present in the process.

The interactions and pervasiveness of physical phenomena make it a challenge to arrive at the required system behaviour of the system-under-development in such a way that it fits the end user's expectation. The central question is how to design this behaviour in the several stages of development that the system goes through. As an illustration, for adaptive systems, graceful degradation can result unexpectedly in an adverse situation, because a user may not immediately attribute the impaired performance to broken parts of the system. So, adaptivity and understandability of the system can have a trade-off here. Actually, this happened in reality in printer development: because a test engineer was not aware of the full set of interactions covered by a newly developed adaptive control loop, he wrongfully attributed an observed printer slow-down, caused by a radiator breakdown, to that adaptive control loop, while the latter was just performing its task perfectly. The control-loop designer obviously did know and could relate the observation to its cause. Thus, it is important to realise that the relationships in the design remain relevant over time: it is a challenge to maintain the knowledge of such relationships throughout all phases of system development.

So, even more than before, system designs are entangled. How then should one take system-level decisions about such adaptive system designs? Is there a systematic, methodological approach for this, apt for the challenge of keeping track of the interdependencies? A basis for a successful method is making insights in relationships and interdependencies explicit and always having them at hand, in multiple views that fit these relations best. This is in our opinion synonymous with having a model-based design approach [24]. In such an approach, models play a pivotal role in coordination within and across development teams, also or especially in the early phases of development: requirements analysis, functional analysis, and design. Note that this fits well with the self-reflection property of adaptive systems mentioned in Sect. 2.3.3.

In the remainder of this section, we focus on aspects of making the middle block in Fig. 2.12 model-based, discussed in three subsections. Firstly, we discuss how models can be specified in a useful manner and what their general structure looks like. Secondly, we discuss how the model specification can serve as the basis for systematic analysis for parts that are quantitatively specified. Lastly, we discuss how model-based specification of system design can also serve as the basis for synthesis, i.e. putting design elements together to form something new, thereby facilitating the implementation of complex systems.

2.4.2 Systematic Modelling in Early Phases

The development of increasingly complex adaptive systems strengthens the need of using models. A model is a simplified representation of the system itself, mostly focused on a few aspects of that system. A model that is able to reconstruct (an approximation of) the operational behaviour of a system is often called an *executable model* (see executable architectures, executable UML [9]). These models play an important role in designing adaptive systems, because in order to design correct and understandable system behaviour it is necessary to reason about dynamic properties of the system. Measurements on prototype systems can either be summarised by relatively simple relationships in such models, or point out problem areas where insights are lacking. System design alternatives can be compared faster and in a more quantitative way using such models. Models can also be at the core of engineering the system, i.e. models can serve as the specification of the system itself, and play a prominent role in system integration and testing.

It is clear that models can and will be used to many different ends. The economical context plays a crucial role: the benefits of using models should be larger than their development and usage costs. Hence, there is a need for a systematic approach for making models, which is very much alike the challenge of structuring a development process as such. Taking this metaphor further, different views with different roles can be discerned, supporting model development by providing separation of concerns. Reusing parts of models, just like subsystems or system components are being reused, is relevant to take into account as well. Integration moments play a driving force in the development process, alike prototype versions of the system that guide the engineering process. One may even claim that physical prototypes are models as well, the only difference being the degree of virtualisation of development. Naturally, models as well as physical prototypes require thorough testing before being used.

To comment on the last step of the sketched model development process: models are only useful when the results they produce harmonise with observations that could have been made on real systems (which also holds for prototypes, although

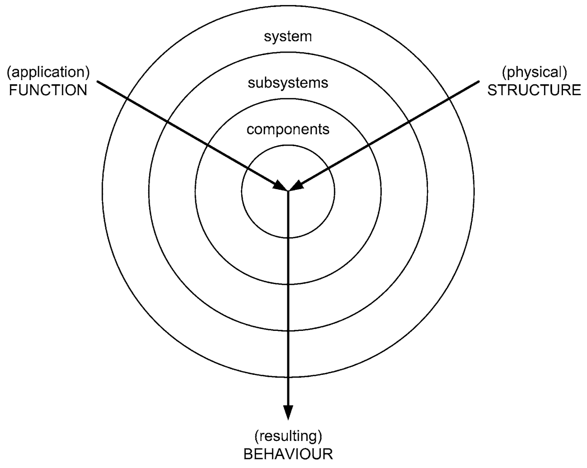


Fig. 2.13 Separation of concerns in modelling: functionality decomposition, structural decomposition, and their relations (mapping) are separately described

the simplifications with respect to the real systems have a different nature). The process to achieve this, so-called *model gauging*, is often underestimated. Often it requires more detailed modelling to deepen the understanding of what is really happening, before the level of simplification can be increased again while being knowledgeable about the constraints and assumptions for which the simpler model still yields trustworthy results.

Separation of concerns is an important means to systematise the modelling process, although over-generalisation is a risk. Especially when a domain is well-defined and well-understood, separation of concerns helps to chart the dimensions of variations for different design options in a structured way. This is applicable for the system overview level as well as the more detailed levels of subsystems and system components. A common way to achieve separation of concerns is sketched in Fig. 2.13.

An example is given by the data path design approach, in which the functional description of the data path is fully separated from the representation of the physical components that implement the data path, and the relationships between these two views is captured in a third view, called mapping. This separation, also known as the Y-chart approach [19], facilitates a better overview of the data path design as such, and exploration of design options is provided in a systematic way. Furthermore, different dynamic analyses of the data path can be performed starting from a single model specification, which makes the risk of having incompatible models very small. Systematic data path modelling for behavioural analysis is the topic of Chap. 7.

At the system level, the challenge to better support system architecting in a systematic way through a model-based approach has been the topic of the research that is described in Chap. 6. Like in the case of the data path, separation of concerns serves as a central starting point: the approach is rooted in the Function-Behaviour-State (FBS) representation scheme [31]. Structural components are synthesised to implement functions, which are derived through detailing a function tree. The behavioural consequences are validated by analyses, either qualitatively or quantitatively, based on physical phenomena that are known about the chosen structural components and taken from a library. The functions, structure, and physical phenomena lead to a parameter network that holds the interrelationships between the design parameters as well as the system qualities. Important system aspects can be evaluated using this parameter network, and design alternatives can be studied in a systematic way.

2.4.3 Cost-Effective Functional Analysis

In the functional analysis phase (see Fig. 2.12), developers perform analyses in order to validate whether the system-under-construction is compliant with the requirements as well as to validate whether it is feasible to realise it. This is an iterative process, progressively including more and more details until all details are filled in, while in parallel the number of open options for the system design is decreasing from virtually infinite to only one in the end, i.e. the system itself. All analysis activities in this phase of the process support such a system evolution. Of course, this is a caricature of reality: often, multiple alternatives are pursued in parallel to reduce development risks. Following the simplified picture of the process, sometimes the scope of design choices is very limited, e.g. when an engineer is detailing a component design to arrive at a concrete realisation. For other design choices, the scope concerns the complete system: the selection of the printing technology (inkjet or toner) affects almost all other decisions.

It is clear that cost-effective analysis of design options is important to construct a beneficial development process. Especially for adaptive systems a model-based approach is beneficial, because the impact of unforeseen design changes is much larger due to the interdependencies of the system's functional parts. Also the impact of unforeseen interactions between system components is much larger, possibly leading to sub-optimal performance or unacceptable system behaviour. With appropriate models, such interactions will already materialise during model development and integration, avoiding costly design iterations involving physical prototypes. A number of factors can be distinguished when thinking about the possible roles of models in the functional analysis phase.

Firstly, a primal factor to consider is the fact that a collection of virtual prototypes (i.e. models) is almost always less costly than the identical collection of physical prototypes. This has its effect on the process: in a process built around physical prototypes developers will strive to make as many design decisions as possible, before validating them with experiments on the prototype. In the case of virtual prototypes, the decision making is much more interwoven with the construction of the prototype. As the models can represent many design variants, experiments do not only serve to validate, but also to explore.

Secondly, the situation is not only black-and-white; grey tones exist as well. Hybrid virtual/physical prototypes, often indicated as X-in-the-loop setups or emulations [34], fulfil an important role in increasing flexibility in the development process. These prototypes enable further decoupling of development of subsystems when their planning does not align very well.

Reuse is an essential ingredient as well. Reuse is not a concept that is limited to implementation only (software code and hardware components), but it also applies to knowledge, to experience, to system-level patterns, etc. Different levels of reuse apply to different phases in development: too early reuse incentives at a too detailed level may lead to ineffective and inefficient solutions. This tension is often observed: a high degree of reuse limits the flexibility of the resulting system [23].

Lastly, computers should be deployed for tasks they can perform very well, which notably holds for repetitive tasks. This means that automated support in exploring possible design alternatives in a large design space, weighing the different options by an explicit evaluation criterion either a priori or a posteriori, can be provided with the aid of a fully parameterised model of (part of) the system. Automated design-space exploration (DSE) for adaptive systems is still in its research phase, but it is an interesting technique to help developers further improve their process. This contrasts with the design of computing hardware components, for which automated DSE is often possible [27]. Chapter 7 shows this in the printing domain for the design of the data path subsystem.

2.4.4 Automated Synthesis in Model-Based Development

As we have illustrated in the previous subsections, the development of adaptive systems is more involved than that of non-adaptive systems. A higher degree of virtualisation of the process, also called *model-based development*, is seen as a means to deal with this. Nevertheless, this is not the only factor that should be considered for adaptive-system development. The process that leads to the engineered in-product software has its own challenges in keeping control of the additional complexity that is introduced by system adaptivity. In fact, the information in the models used by developers to reason about the system, should be present in the self-reflection properties of the system, indicated in Sect. 2.3.3, and these have to be implemented in the in-product software.

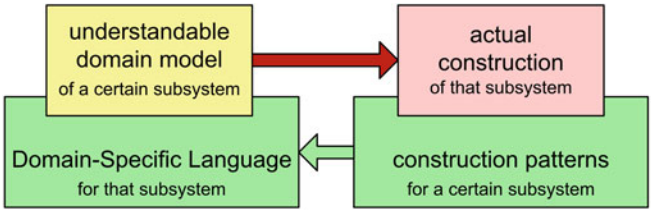


Fig. 2.14 The basic principle of domain-specific languages and code generation. *Language* is synonymous to *formalism* in this scheme. *Patterns* include *templates* for software construction as a particular realisation of this scheme

Functional analysis is often done by other developers than those involved in system synthesis. In current practice, the design resulting from functional analysis is the starting point for implementation, which implicitly means that the engineers in this phase have to perform a double interpretation: what is meant with the design, and how shall it be implemented? The separation of these two steps, i.e. the problem specification and synthesised solution, together with automated translations from specification to implementation, are the basic ingredients for a popular approach in software engineering: code generation using *domain-specific languages* (DSLs, see Fig. 2.14). Again the main principle is separation of concerns: the specification can be made accessible to a wider group of developers who should be concerned with solving the problem in a functional, domain-specific way, while the software-implementation-specific choices are captured in an orthogonal way, i.e. not repeating a large part of the specification effort, but rather focusing on correct translation of the specification into an implementation.

In Chap. 8 this topic is explained along the lines of two examples from the printing domain: job scheduling and multi-objective optimisation focused on control software. In each of these cases a domain-specific language enables the formulation of concise and understandable models that can be transformed into implementations in a manageable way. This not only supports the communication between system and software developers, it also supports dealing with evolving requirements and insights in the respective domains over time.

2.5 Organisation: How to Realise the Process?

The last view of the BAPO framework considers the people and organisational structures that are in place to execute the system development. Ideally, the organisational aspects should follow the previous views, but practice shows that the current

Organisation always imposes strong constraints on Business, Architecture, and Process. And rightfully so, because the developers' competences cannot be changed overnight to follow a completely different process. In this section, we discuss organisational structure as well as engineering competences.

2.5.1 Development Project Structure

The organisational challenge that is put forward by the development of adaptive systems can be seen as an evolutionary step in the organisation of complex system development. Not all companies organise their development in the same way: some choose engineering disciplines as the main factor to form different teams, others choose subsystems as the main discerning factor. Either way, an organisational decomposition is followed in order to facilitate the division of responsibilities, work breakdown, and confinement of the required communication between different people (see [23]).

Ultimately, the behaviour at system level is relevant to all stakeholders, notably those outside the development team. Specific measures, such as project roles, are often in place to guard system properties and behaviour. Most conspicuous in this respect are the architect roles that are essential in system development. Architects shape and guard system aspects; for complex systems often there is an architect team, each member of which takes care of a number of system aspects.

With the advent of adaptive systems *pur sang* the need for explicitly managing cross-links between different subsystems becomes more important. This is a trend that we have seen in the course of the Octopus project as well, although no research has been conducted in this area. Nevertheless, the balance between subsystems engineering and the architect team has shifted: it has been observed that in early project phases for adaptive systems the cross-linking architecting roles tend to be more strongly emphasised than before.

2.5.2 Competence Shift for DSL-Based Software Engineering

The need to deal with increased interactions between subsystems has led us to discuss domain-specific languages in Sect. 2.4.4. The DSL-based engineering approach has a significant effect on organisational aspects as well, mainly visible in the changed responsibilities for the software engineers.

The DSL approach introduces two tasks that are relatively new. These tasks are directly related to the core principles of the approach. First, it is the primary goal of DSLs to specify (partial) designs at the *what* level in a specific language that is built with familiar domain concepts. The benefit is that engineers can share them already in the functional analysis phase without involving implementation details. On the other hand, such language(s) should be constructed and maintained.

Second, the focus of software engineering shifts from regular software construction to the development of *transformation mechanisms* to generate code from DSLs. These transformations are similar to those in classic compilers, but whereas compilers mostly are quite generic and domain-independent software programs, the implementations of the transformation mechanisms are inherently domain-specific. Hence, additional development and maintenance tasks are introduced with DSL-based software engineering.

Cost-effective realisation of the two tasks mentioned in the previous paragraph is an essential factor for the successful introduction of DSL-based software engineering. In our view, this can only be achieved in areas where the problem domain is shared across a range of products and the proposed design solutions employ similar patterns. Fulfilling this premise comes in its turn very close to having a *reference architecture* in place, in which – within a certain domain, or certain (sub)system – a limited set of solution patterns is applied to generate a complete (sub)system.

Following this line of reasoning, two interrelated groups of engineers are needed in DSL-based software engineering: one that works on (reusable) specification formalisms, which are understandable for a broader community, and another one that works on transformations, biased towards reuse over different products. In Océ, these two groups can indeed be observed in the area of the real-time embedded software, which is governed by ESRA (embedded software reference architecture) [11]. One of these groups works on uniform specifications in a subset of UML-RT, and the other group, a reuse group, plays an important role in developing and maintaining libraries that support software realisation from the aforementioned specification in UML-RT [7].

2.5.3 DSL-Based Modelling

Models of behaviour have quite a degree of similarity with software; actually, also their development is much alike software development. Therefore, much of the reasoning in the previous section about software holds for models as well. This becomes all the more clear when one realises that modelling formalisms are often as complex as software (or even more so), and fewer experts are to be found in industrial contexts for a range of modelling formalisms. A way to address this challenge is to specify models in better-known formalisms and transform those specifications to the models that are really needed. Still, the maintenance of the transformations would be a newly introduced task, which is expected to be limited as the formalisms are in general quite stable over time. It should be noted that this task demands highly scarce competences regarding knowledge of both the formalisms and product-related modelling requirements. Explicit organisational measures are necessary to guard availability when needed.

In the Octopus project, two research activities can be seen along this view.

First, the data path modelling discussed in Chap. 7 separates model specification from the actual models to analyse data path performance. This research illustrates very well that the specification can be shared across a larger group of stakeholders without bothering them with the intrinsic difficulties of applying model formalisms like timed automata [2], coloured Petri nets [17], or synchronous dataflow [29].

Second, the topic of specification of Bayesian networks [16] starting from other formalisms is studied in Chap. 5. The coupling of electrical circuit theory to Bayesian networks indicates that these involved formalisms are far from independent. Explicit construction of a domain-specific language that is understandable, but also prepares for the implementation constructs, is an easier route to go than adaptation of known formalisms that are too far apart from each other. The key issue is again to find a connection between a DSL and a formalism. If the connection carries too many drawbacks like dependencies of formalisms, constrained expressiveness, or difficult transformations, a better usable connection has to be searched for.

2.6 Summary and Conclusion

In this chapter we have given a high-level account of adaptive systems. Assisted by a structuring of the discussion through the BAPO aspects of Business, Architecture, Process, and Organisation, and by examples from the domain of professional printing, the effects of increasing the level of adaptivity of already complex systems have been discussed.

In summary, the main organisational issues are related to the fact that skills and knowledge are not changed overnight. In processes, new development paradigms induce evolution of development processes that integrate new ways of working in which models and modelling are at the basis of acquiring and sharing knowledge, and exploring the design options. The architectural challenges address the need to find proper choices of methods, formalisms, and transformations for effective and efficient architecting of adaptive systems. Explicit representation of information and its uncertainty about the system and the system's context is a major ingredient for the architectures of adaptive systems. These all add to the business goal of deploying the impact of fast changes of product requirements from the market in adaptations of the architecture, processes, and organisation.

From this global summary, two main categories of the research that has been conducted in the Octopus project have been discerned: one related to architecture and solution patterns for adaptive systems, the other related to the development process of adaptive systems. It goes without saying that the research topics of the Octopus project cover neither complete architectures nor complete development processes. It was even not the intent of this chapter to sketch the complete field of adaptive systems. Ideally, the reader now has an impression of how the research contributions fit together in a logically coherent over-arching storyline of adaptive systems, illustrated in the field of professional printing.

Many challenges still remain, and will remain for a considerable time. It is our intent that framing the research issues in an industrial context will help weighing their respective relevances. Cost/benefit ratios of research topics are often not immediately clear or positive in such a context. Overseeing the adaptive systems engineering field from the point-of-view of the Octopus project we claim that many more research projects with validation in industrial contexts are needed to advance the development of adaptive systems with state-of-the-art methodologies. These projects may benefit from the lessons learned in Octopus which are laid out in this book.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. Agent iSolutions: http://www.agentisolutions.com. Accessed Jun 2012
2. Bengtsson, J., Yi, W.: Timed automata: Semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets. Lecture Notes in Computer Science, vol. 3098, pp. 87–124. Springer, Berlin (2004)
3. Boehm, B.W.: A spiral model of software development and enhancement. SIGSOFT Softw. Eng. Notes **11**, 14–24 (1986)
4. Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J., Kanade, T.: Footstep planning for the Honda ASIMO humanoid. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), Barcelona, pp. 629–634 (2005)
5. Cogniteam: http://www.cogniteam.com. Accessed Jun 2012
6. de Roo, A.J.: Managing software complexity of adaptive systems. Ph.D. thesis, University of Twente, Enschede (2012)
7. Dohmen, L.A.J., Somers, L.J.: Experiences and lessons learned using UML-RT to develop embedded printer software. In: Oivo, M., Komi-Sirviö, S. (eds.) Product Focused Software Process Improvement. Lecture Notes in Computer Science, vol. 2559, pp. 475–484. Springer, Berlin (2002)
8. Eisner, H.: Essentials of Project and Systems Engineering Management. Wiley, Hoboken (2008)
9. Estefan, J.A.: Survey of model-based systems engineering (MBSE) methodologies. Technical Report INCOSE-TD-2007-003-01, INCOSE, San Diego (2008)
10. Fitzgerald, B.: Formalized systems development methodologies: A critical perspective. Inf. Syst. J. **6**, 3–23 (1996)

11. Geelen, H.: Reference architecture from management perspective. MOOSE seminar, Helsinki (2004)
12. Gheorghita, S.V., Palkovic, M., Hamers, J., Vandecappelle, A., Mamagkakis, S., Basten, T., Eeckhout, L., Corporaal, H., Catthoor, F., Vandeputte, F., de Bosschere, K.: System-scenario-based design of dynamic embedded systems. ACM Trans. Des. Autom. Electron. Syst. **14**, 3:1–3:45 (2009)
13. Henzinger, T.A., Sifakis, J.: The embedded systems design challenge. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006: Formal Methods. Lecture Notes in Computer Science, vol. 4085, pp. 1–15. Springer, Berlin (2006)
14. Hobbelen, D., de Boer, T., Wisse, M.: System overview of bipedal robots Flame and TUlip: Tailor-made for limit cycle walking. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2008 (IROS 2008), Nice, pp. 2486–2491 (2008)
15. Hoffmann, H., Maggio, M., Santambrogio, M.D., Leva, A., Agarwal, A.: SEEC: A framework for self-aware computing. Technical Report MIT-CSAIL-TR-2010-049, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA (2010)
16. Jensen, F.V.: An Introduction to Bayesian Networks. UCL, London (1996)
17. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, Berlin (2009)
18. Kemperman, J.E.B., van Engelen, M.L.: Operationalizing the customer value concept. In: Proceedings of the 28th EMAC Conference: Marketing and Competition in the information age, Berlin (1999)
19. Kienhuis, B., Deprettere, E., Vissers, K., van der Wolf, P.: An approach for quantitative analysis of application-specific dataflow architectures. In: Proceedings of the 1997 IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '97), Zurich, pp. 338–349 (1997)
20. Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. Addison-Wesley Pearson Education, Inc., Boston (2004)
21. Martín H., J.A., de Lope, J., Maravall, D.: Adaptation, anticipation and rationality in natural and artificial systems: Computational paradigms mimicking nature. Nat. Comput. **8**, 757–775 (2009)
22. Mathijssen, R. (ed.): Trader: Reliability of High-Volume Consumer Products. Embedded Systems Institute, Eindhoven (2007)
23. Muller, G.: Systems Architecting: A Business Perspective. CRC, Boca Raton (2012)
24. Nossal, R., Lang, R.: Model-based system development: An approach to building X-by-wire applications. IEEE Micro **22**, 56–63 (2002)
25. Papalambros, P.Y., Wilde, D.J.: Principles of Optimal Design: Modeling and Computation. Cambridge University Press, Cambridge (2000)
26. Rao, A.S., Georgeff, M.P.: BDI-agents: From theory to practice. In: Proceedings of the First International Conference on Multiagent Systems (ICMAS'95), pp. 312–319 (1995)
27. Saxena, T., Karsai, G.: MDE-based approach for generalizing design space exploration. In: Petriu, D., Rouquette, N., Haugen Ø. (eds.) Model Driven Engineering Languages and Systems. Lecture Notes in Computer Science, vol. 6394, pp. 46–60. Springer, Berlin (2010)
28. Sommerville, I.: Software Engineering, 3rd edn. Addison-Wesley, Wokingham (1989)
29. Stuijk, S., Geilen, M., Basten, T.: SDF^3: SDF For Free. In: Proceedings of the Sixth International Conference on Application of Concurrency to System Design (ACSD 2006), Turku, pp. 276–278 (2006)
30. The Office of the Deputy Assistant Secretary of Defense for Systems Engineering: http://www.acq.osd.mil/se/. Accessed Jun 2012
31. Umeda, Y., Kondoh, S., Shimomura, Y., Tomiyama, T.: Development of design methodology for upgradable products based on function-behavior-state modeling. Artif. Intel. Eng. Des. Anal. Manuf. **19**, 161–182 (2005)
32. van de Laar, P., Punter, T. (eds.): Views on Evolvability of Embedded Systems. Springer, Dordrecht (2011)
33. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer, Berlin (2007)

34. Visser, P.M., Groothuis, M.A., Broenink, J.F.: Multi-disciplinary design support using hardware-in-the-loop simulation. In: Proceedings of the 5th PROGRESS Symposium on Embedded Systems, Nieuwegein, pp. 206–213 (2004)
35. Winikoff, M.: JACK^{TM} intelligent agents: An industrial strength platform. In: Bordini, R., Dastani, M., Dix, J., El Fallah Seghrouchni, A., Weiss, G. (eds.) Multi-Agent Programming. Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 15, pp. 175–193. Springer, New York (2005)
36. Zadeh, L.A.: Optimality and non-scalar-valued performance criteria. IEEE Trans. Autom. Control **8**, 59–60 (1963)

Chapter 3
Piezo Printhead Control: Jetting Any Drop at Any Time

Sjirk Koekebakker, Mohamed Ezzeldin, Amol Khalate, Robert Babuška, Xavier Bombois, Paul van den Bosch, Gérard Scorletti, Siep Weiland, Herman Wijshoff, René Waarsing, and Wim de Zeeuw

Abstract Full flexible use of inkjet printhead units in printing systems requires consistent generation of drops with any given volume and velocity at any moment and place desired. True drop-on-demand is currently hampered by physical phenomena in the printhead. These are residual vibrations and crosstalk resulting from conventional jets. This chapter presents control strategies to overcome these problems. First, with experiment-based control the drop characteristics are measured and the jet pulse that activates the jetting of a drop is optimised. Choosing a proper jet pulse structure, one can deal with single-channel residual vibration, multi-channel crosstalk, and even generalise optimisation over each bitmap to be printed. Secondly, with a model-based control approach, optimised jet pulses can be derived without additional measurement equipment. Considering the inkjet mechanism as an uncertain system and designing a robust pulse allows to deal with differences

S. Koekebakker (✉) • H. Wijshoff • R. Waarsing • W. de Zeeuw
Océ-Technologies B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands
e-mail: sjirk.koekebakker@oce.com; herman.wijshoff@oce.com; rene.waarsing@oce.com; wim.dezeeuw@oce.com

M. Ezzeldin • P. van den Bosch • S. Weiland
Control Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: m.ezz@tue.nl; p.p.j.v.d.bosch@tue.nl; s.weiland@tue.nl

A. Khalate • R. Babuška • X. Bombois
Delft Center on Systems and Control, Faculty of Mechanical, Maritime and Material Engineering, Delft University of Technology, P.O. Box 5, 2600 AA Delft, The Netherlands
e-mail: a.a.khalate@tudelft.nl; x.j.a.bombois@tudelft.nl; r.babuska@tudelft.nl

G. Scorletti
Laboratoire Ampère, Ecole Centrale de Lyon, 36 Avenue Guy de Collongue, 69134 Ecully Cedex, France
e-mail: gerard.scorletti@ec-lyon.fr

T. Basten et al. (eds.), *Model-Based Design of Adaptive Embedded Systems*, Embedded Systems 22, DOI 10.1007/978-1-4614-4821-1_3,
© Springer Science+Business Media New York 2013

between model and real system. Both the experiment- and model-based method result in strongly improved drop characteristics, which is experimentally verified and thereby provide very valuable steps towards adaptive printing systems.

3.1 Motivation

In colour inkjet print engines, the central component is its imaging device, the printhead. In this chapter, a systems and control approach is taken towards achieving optimal operation of this core part of one of the most significant print technologies of today and the future. Motivation for this approach can be found in [8, 12, 15].

In a wider perspective, a strong industrial motivation for the research presented in this chapter can be read in the recently published roadmap to printing as part of the Dutch High-Tech Systems and Materials platform [25]. The graphical printing industry globally turns over more than 500 billion dollar a year. Although only 5% of this market consists of digital printing now, this part of the industry is expected to keep on growing with double digits in the years to come. Inkjet is becoming the dominant printing technology in this market. The transition from both analogue to digital and from black-and-white to colour printing, next to emphasis on cost effectiveness, are the main drivers in this development. Ever-increasing speed and cost-effective improvement of reliability are seen as the main technological challenges in this market.

Parallel to these transitions, yet another even faster growing market, that of digital fabrication is believed to take off. Also here inkjet is believed to become one of the core enabling technologies in this digital industrial revolution where printing of information will be generalised to printing of things. Inkjet technology here offers highly efficient use of materials as it is an additive process, is contact-free with respect to the substrate to be printed on, and enables accurate deposition of various materials in complex patterns. In industrial printing of functional materials, the requirements with respect to speed, accuracy, and reliability will often be more tight compared to the graphical printing market. In this important fast-growing industry it is very worthwhile to perform research into better control over its key component, the inkjet printhead.

Increased demands on versatility in the use of the inkjet technology will be required. Complete control over the size, velocity, and jet timing is desired for every drop. This has to be attained over a large latitude in a fully adaptive printing system with variable-speed printing. Complete control over deposition means high quality combined with the ability to tune the printer for each application, highly productive if the interaction between substrate and ink allow this, and tolerant to fast-varying jet frequencies resulting from more complex substrates or low-cost engines. The challenge tackled in this chapter was even larger. With a given printhead as constraint, it was investigated to what extent this goal can be attained by using smart feedforward control techniques.

The core component in the inkjet printer is the printhead. This imaging device is the enabler in the marriage of ink and substrate resulting in the final product.

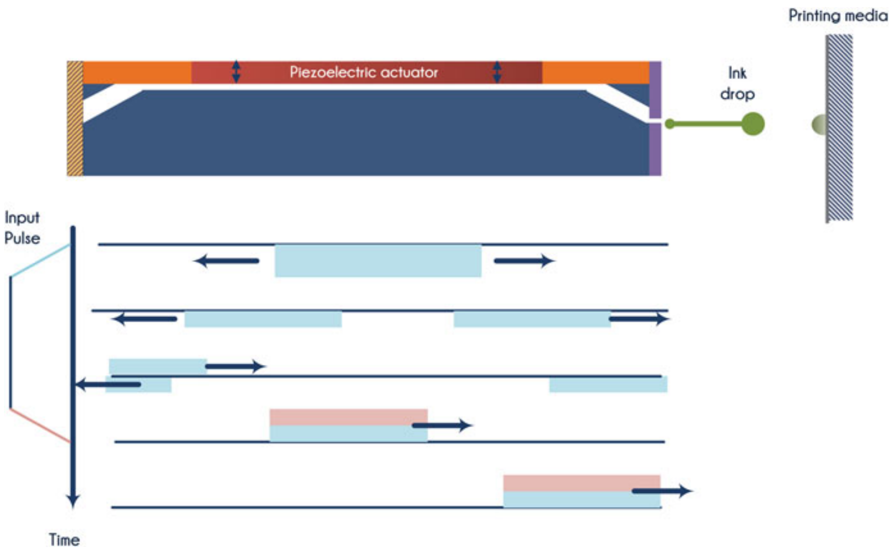


Fig. 3.1 Drop jetting mechanism

Various different inkjet technologies exist but the piezo-driven printheads are generally seen as the most reliable and most versatile with respect to the materials to be jetted. Typical reliability numbers are e.g. a billion jets without failure for each inkjet channel and the ability to give a trillion (10^{12}) jets over the life-time thereby jetting tens of kg's of ink.

As drops have to travel after being jetted before they hit the substrate, velocity variations inevitably lead to positioning errors as in printing there is almost always relative motion between printhead and substrate. Wrongly positioned drops easily result in print artifacts in graphical applications or even product failure in digital fabrication. To put accuracy of an inkjet printhead in perspective, consider a nominal drop velocity of 5 m/s and between the substrate and image device a relative velocity of 1 m/s and a distance of 1 mm. Now every error in jet timing of 10 µs will result in a positioning error of 10 µm of the drop on the substrate and an error of 0.5 m/s in drop velocity will lead to a drop positioning error of 20 µm. These positioning errors should typically not exceed 10 µm.

The basic operation of a piezo-driven printhead ink channel is given in Fig. 3.1. Most printheads consist of several (typically hundreds) of separate ink channels. In the top part of Fig. 3.1, one jetting ink channel is shown schematically. This channel has an open connection on the left to a reservoir (through a filter), and on the right through the nozzle, a small opening to the outside world acting as a closed connection. The pressure in each channel can be influenced by an individually driven piezo element (top). The piezo element uses the acoustics of the channel to efficiently jet a drop. First a negative pressure wave is introduced by piezo retraction. This pressure wave over the channel length at several time instances is shown in

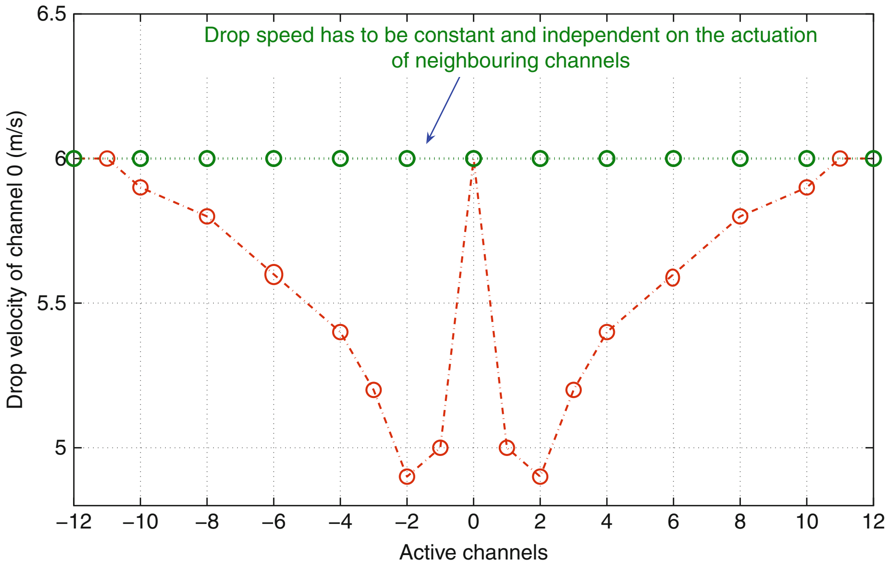


Fig. 3.2 Crosstalk: Influence on the drop velocity of channel 0 (*centre* channel) by actuation of different neighbouring channels at the same time

light blue beneath the drawing of the printhead. On the left the corresponding piezo input is given. This pressure wave will reflect at the nozzle and the reservoir sides. The part which reflects at the reservoir, which acts as an acoustic open end, will travel as a positive wave towards the nozzle. This positive pressure wave travelling to the nozzle is doubled by piezo extension to the original state and at the same moment the wave which travelled to the reservoir is cancelled. The pressure wave travelling to the nozzle enforces the drop formation.

Ideally the channel should be at rest after this drop formation. In practice two problems occur. First, there will be residual vibrations. Secondly, there can be crosstalk between the neighbouring channels. This influence is shown in Fig. 3.2. In practice the crosstalk is often minimised by using a delay between neighbouring channels. The residual vibration, shown in Fig. 3.3, can become a problem when moving to a higher productivity. The figure shows the velocity of the meniscus, i.e. the interface between the ink and air at the nozzle, over time due to the described piezo input, the standard pulse u_{std}. The drop will be emitted after circa 9 μs. The meniscus position will be discontinuous at this point, its velocity is not in this model. The figure shows that drops jetted from a specific channel can only be jetted interaction free at very low ($<$10 kHz) jet frequencies using the standard pulse. Productive printing requires higher jet frequencies with controllable drop properties.

The printhead has to jet any random bitmap. Best quality can be achieved only when all drops have the same properties. It means drop properties should be independent of the frequency at which the drops are demanded and the specific

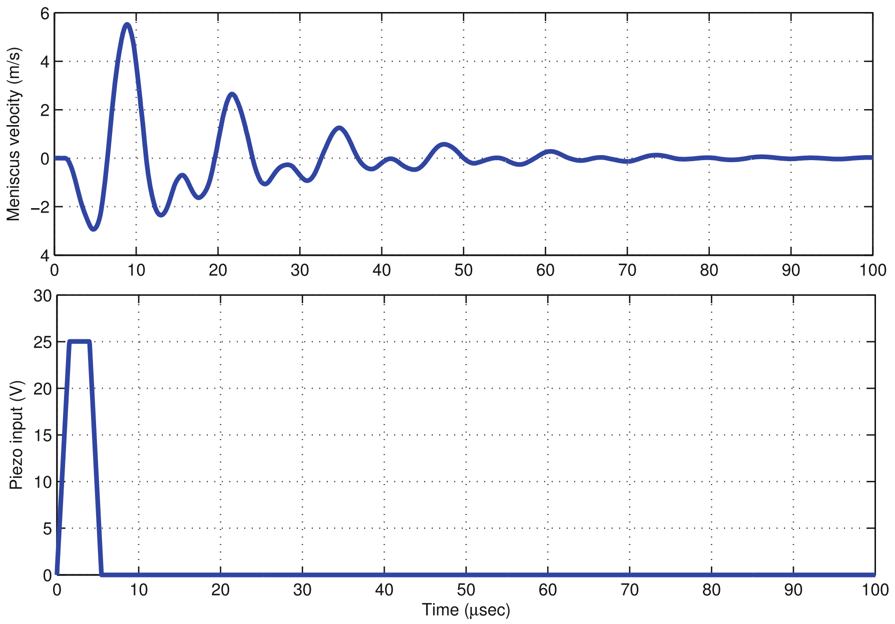


Fig. 3.3 System response for the standard pulse, u_{std}

bitmap. With the standard pulse, the drop velocity is influenced by the drop-on-demand (DoD) frequency. The residual vibrations result in variable drop velocity while jetting at several fixed jetting frequencies. This is shown in Fig. 3.4. Here the DoD curve is given as the drop velocity of a continuous jetting stream as a function of frequency, i.e. only steady-state drop properties are plotted. This is attained generally after jetting 5–6 drops continuously.

Even though ultimately every drop needs to be the same, the first step towards achieving this goal can be to flatten this DoD curve. This will ensure that at least in steady-state drop properties will be independent of the DoD frequency. In industry often a pragmatic approach is taken. A nominal jet frequency is chosen such that this frequency and its integer divisions result in the same drop velocity. The nominal jet frequency will occur with only ones in the bitmap. The integer division will play a role when zeros are part of the bitmap to be printed. With a printhead showing the characteristics of Fig. 3.4, this could be 59 kHz where 29.5 kHz also leads to ca. 4.5 m/s drop velocity.

A flat DoD curve will be necessary but not sufficient for fully adaptive printing. As defined, the DoD curve is taken for fixed operation points here. Transients will result by switching over or moving through different frequencies for this non-linear dynamic system. Improved control over the drop properties requires taking into account these system characteristics. This can be done by modelling and/or performing experiments. From these measurements or models, the proper information has to be extracted to be able to calculate a somehow optimal input pulse.

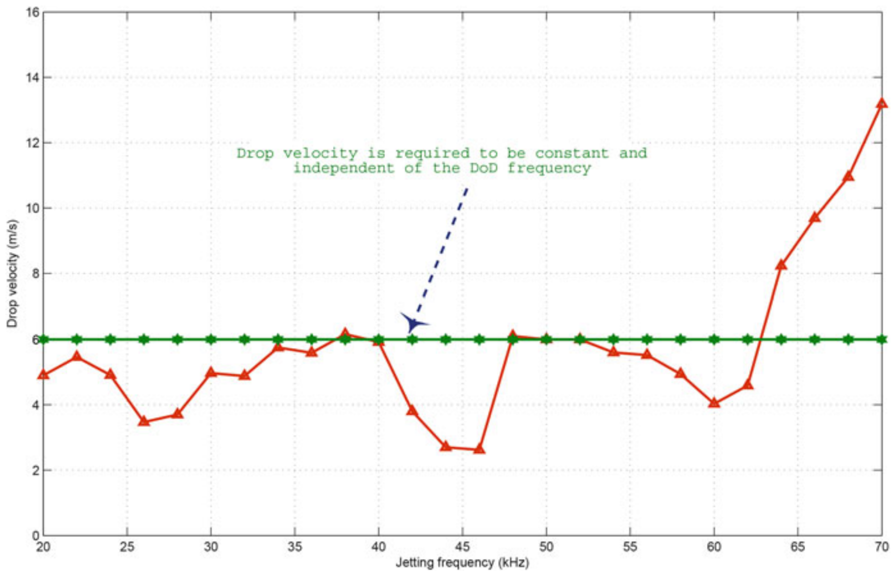


Fig. 3.4 Drop-on-demand velocity curve for the standard pulse

This chapter is built up as follows. In the next section first printhead characteristics and modelling of such a system are discussed. Then a general introduction to a systems and control approach for optimal feedforward is given in Sect. 3.3. Two lines of approach to feedforward design of a printhead are presented. First, feedforward design for the printhead using experimental data is discussed in Sect. 3.4. Then a model-based approach including uncertainty modelling is presented in Sect. 3.5. Both treat the single-channel situation dealing with residual vibration. The extension towards the multi-channel control problem with crosstalk and dealing with bitmaps is given for the experiment-based approach in Sect. 3.6. Finally a discussion of the results and conclusions is given in Sect. 3.7.

3.2 Piezo Printhead Modelling

Modelling of the piezo printhead can become very complicated. For control system design, a model that has less complexity with a reasonable accuracy is required. Limited complexity is required to be able to optimise and synthesise controllers. Some inaccuracy is acceptable by using robust design techniques.

Several analytical and numerical models, that describe the dynamics of the ink channel, are available in the literature [24, 30]. In general, the numerical models are very accurate and they include a high level of details. These models are usually finite element models in which the governing differential equations are numerically solved using a complex mesh and other solving techniques. The main drawback of these models is that they require a very high computational effort, at least several hours to jet one drop. That makes them not suitable for control design. On the other hand, analytical models are less complex since the governing differential equations are simplified to be solved analytically. Sometimes over-simplification results in models with poor accuracy while under-simplification leads to models with high complexity. Combined models that include both numerical and analytical models result in models with less computation time with a reasonable accuracy.

In this section, we give a short overview about piezo printhead modelling. Based on the model complexity, we present three different modelling strategies. First a lumped-parameter model is discussed, where it is assumed that each element of the printhead can be modelled as a lumped element. After that we present a two-port model, which is derived using energy exchange principles. Finally, to gain more insight into the channel dynamics, a fully detailed model based on the so called narrow-gap theory is discussed.

3.2.1 Lumped-Parameter Model

An often used lumped-parameter modelling approach adopts an equivalent electrical circuit to describe the dynamics of the ink channel [3]. This modelling technique is a useful and commonly applied analysis approach for designing piezoelectric inkjet systems. In this modelling framework, resonances are modelled using capacitors, resistors, and inductors in series. Additional resonances are included by placing additional capacitor-resistor-inductor sets in parallel. The inductance represents the inertia, which is related to the fluid mass, the capacitance is a measure of fluid energy storage, volume, and elasticity, and the resistance is associated with any losses causing energy dissipation in the fluid, typically viscous losses. Accurate determination of the equivalent circuit parameters requires either a physical prototype or an accurate computational model to provide the data. In this model, it is assumed that the characteristic wavelength is larger than all dimensions in a channel. That means that the fluid in the flow direction is uniform at any instant in time. This assumption implies that the time required to transmit a change in the applied electric field on the piezoelectric actuator to a change in the meniscus shape at the nozzle is negligible. This is not always the case and delay lines can play an important role. In [3], a transmission-line model of the piezoelectric actuator, fluid chamber, and nozzle is proposed. This model permits analysis without the wavelength limitation and provides the ability to analyse the approximate interaction between multiple resonances in the complete system. However, solving a lumped-parameter model with transmission lines is far from trivial.

3.2.2 Two-Port Model

In [12], a two-port model is proposed to describe the dynamics of the ink channel. This model employs the concept of bilaterally coupled systems. The ink channel is divided into subsystems, namely reservoir, piezoelectric actuator, channel, connection, and nozzle. Each subsystem is modelled as a two-port system based on first-principle modelling. The two-port model of an ink channel is obtained by connecting the subsystems and applying the boundary conditions. This model has less complexity and requires little computational time. However, due to modelling discrepancies, this model cannot accurately capture the first resonance frequency of the channel dynamics, which is the most important resonance frequency. Moreover, although a coupling technique was proposed, the available model does not consider the interactions between an ink channel and its neighbours yet.

3.2.3 Narrow-Gap Model

The narrow-gap model, proposed by Wijshoff [30], describes the dynamics of an ink channel based on the narrow-channel theory [2, 27]. This model describes the dynamics of one ink channel from the piezoelectric input voltage to the meniscus velocity. It also describes the interactions between one channel and neighbouring ink channels from the piezo input voltage to the meniscus velocity.

The derivation of the narrow-channel equations is based on the Navier-Stokes equation of motion and the continuity equation. The continuity equation results from the balance between the change in mass and the flux of mass in a control volume. The Navier-Stokes equation, which is the continuous version of Newton's second law, relates the inertial acceleration of particles of a fluid with internal and external forces that affect the channel.

In the narrow-gap model, the frequency response of the system is obtained via a convolution in the frequency domain, which is equivalent to the swept-sine technique. The frequency response of this model at a frequency ω_0 is computed based on solving the wave equations for a sinusoidal input with the same frequency ω_0. The frequency response is computed by repeating the same procedure over a frequency range. The frequency response of the printhead is obtained as shown in Fig. 3.5. The detailed derivation of this model is given in [30]. This model has a relatively low complexity. This model was validated with laser doppler measurements of the meniscus movement. We assume that this model represents the dynamics of the ink channel with a reasonable accuracy.

To summarise, there is always a trade-off between model complexity and accuracy. Modelling of the piezo printhead can become very complicated. For control system design, a less complex model with a reasonable accuracy is required. The narrow-gap model fulfils these requirements well compared with the other available models. Therefore, throughout this chapter we use the narrow-gap model as a representation of the printhead dynamics.

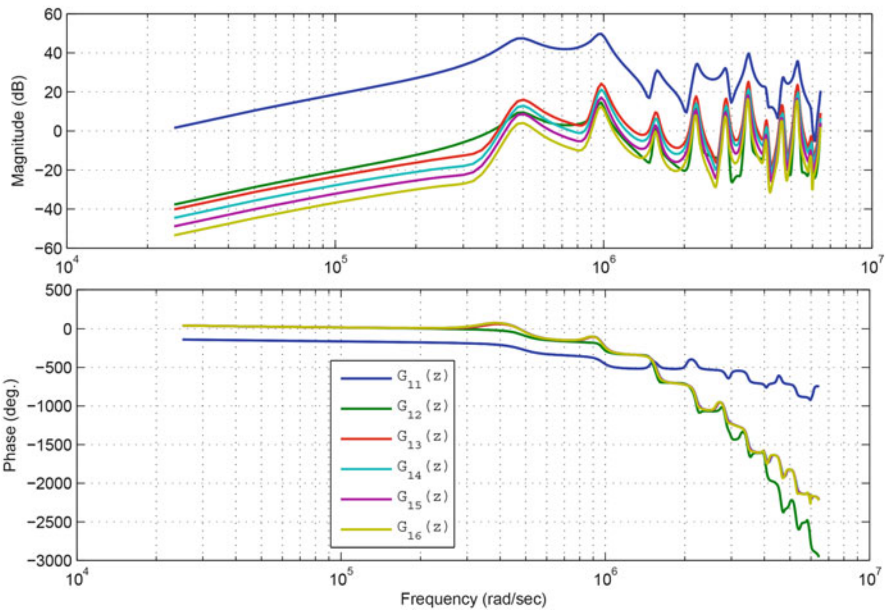


Fig. 3.5 Frequency response of the narrow-gap model for an ink channel, G_{11} and the influence from its neighbours, G_{1i}

3.3 Input Design Optimisation

The drop consistency of a DoD inkjet printhead can be optimised using a systems and control approach. Recently such process performance optimisation has received great attention, since it represents the natural choice for reducing production costs, improving product quality, and meeting safety requirements and environmental regulations. Optimisation is typically based on a model that is used by a numerical algorithm to compute the optimal solution.

The printhead under investigation has limited control capabilities. It is difficult to use feedback control due to the following limitations of the actuation system:

- No sensor is provided for real-time measurement of the channel pressure or of the meniscus position/velocity.
- The sample time required for control computation must be very short due to high drop jetting (DoD) frequencies.
- The driving electronics limit the range of the actuation pulses that can be generated in practice.

In this scenario, the ink channel dynamics can be controlled using a feedforward strategy.

3.3.1 Control Using a Feedforward Strategy

Before discussing control of an inkjet printhead, it is important to investigate in general how feedforward control can be used for input design. In this section, we present a simple approach to (re-)design the input actuation pulse using feedforward control. For optimisation and implementation, it is most convenient to work in the discrete-time domain. At each sample k we have the input piezo actuation, $u(k)$ and the output meniscus velocity, $y(k)$. We need a model of the system to be controlled. Let the discrete transfer function $H(q)$ describe the dynamics from u to y where q is the one sample forward shift operator. We have discussed in the introductory section that the drop properties are mainly governed by the meniscus velocity. Therefore, if we ensure a similar meniscus velocity profile for all jetted drops jetted then we can expect the properties of the jetted drops to be similar as well. Hence, if we know the reference meniscus velocity profile $y_{\text{ref}}(k)$, which the inkjet system $H(q)$ should track to jet a drop of the desired properties, then the optimal control input can be obtained as the one minimising the tracking error $e(k) = y_{\text{ref}}(k) - y(k)$. More precisely, the optimal control input has to minimise the objective function, which can be defined as the sum of squared errors:

$$\mathscr{J} = \sum_{k=0}^{N} \Big(y_{\text{ref}}(k) - y(k) \Big)^2 = \sum_{k=0}^{N} \Big(y_{\text{ref}}(k) - H(q)u(k) \Big)^2, \tag{3.1}$$

where $N = \frac{T}{T_s}$, T_s is the sampling time, T is the final time, $H(q)$ is the discrete-time transfer function, and q is the forward shift operator.

In literature, many frequency-domain and time-domain methods are proposed to obtain the feedforward control input. One of the popular frequency-domain methods is dynamic inversion wherein the inverse of the system model $H(q)$ is used to obtain the feedforward input. However, when the system $H(q)$ is non-minimum phase and strictly proper (i.e. order of the numerator polynomial is lower than the denominator), use of this method is far from trivial. The meniscus response to a actuation pulse has non-minimum phase characteristics as the meniscus position first drops before it peaks in a positive direction, see Fig. 3.3. In such a scenario, one can use a time-domain method such as model predictive control [7, 26]. Nevertheless, when the control sequence to be computed is long, this method is computationally expensive. We present a simpler and computationally efficient filtering-based approach to generate the unconstrained actuation pulse proposed in [16]. For this purpose, we parameterise the to-be-designed actuation pulse as the pulse response of a Finite Impulse Response (FIR) filter $F(q,\beta)$:

$$u(k,\beta) = F(q,\beta)\delta(k), \tag{3.2}$$

where $F(q,\beta) = \beta_0 + \beta_1\ q^{-1} + \ldots + \beta_{n_\beta} q^{-n_\beta}$, $\delta(k)$ is the unit pulse, and $\beta = (\beta_0, \ldots, \beta_{n_\beta})^T$ is a vector containing the coefficients of the FIR filter. When the dimension of β is chosen equal to the desired length of the input waveform, this parametrisation allows to generate an unconstrained input sequence.

For an arbitrary vector β, the response of the ink channel $H(q)$ to the input $u(k,\beta)$ is given by:

$$y(k,\beta) = H(q)F(q,\beta)\delta(k) = F(q,\beta)H(q)\delta(k) \tag{3.3a}$$

$$= F(q,\beta)h(k) \tag{3.3b}$$

where $h(k)$ is the pulse response of the known ink channel dynamics $H(q)$.

As discussed earlier, the optimal control $u(k,\beta_{\text{opt}})$ is the one which minimises the objective function (3.1). Therefore, the optimal parameter vector β_{opt} is the solution of the following optimisation problem:

$$\min_{\beta} \sum_{k=0}^{N} \left(y_{\text{ref}}(k) - F(q,\beta)h(k)\right)^2. \tag{3.4}$$

Note that the optimisation problem (3.4) leading to the unconstrained input pulse is a linear least-squares problem. When this problem (3.4) is numerically ill-conditioned, it is advisable to approximately solve (3.4) using a truncated Singular Value Decomposition (SVD) (for more details, see [10]).

For simplicity, we have presented here only the single-input single-output (SISO) case. The extension of this method to the multi-input multi-output (MIMO) case is straightforward. Using the MIMO model of the inkjet printhead, one has to design a FIR filter for each ink channel such that each ink channel follows its reference trajectory, for details see [18].

3.3.2 Dealing with System Uncertainty Optimising Performance

An accurate model of complex processes can be hard to find in practice with reasonable effort. The system identification is further complicated, especially in the case that the system measured data are noisy and in dealing with signals, which often do not have sufficient information for efficient system identification [28]. Therefore, optimisation using an inaccurate nominal model only may result in a sub-optimal solution or, even worse, infeasible solution when constraints are present.

An important part of the model uncertainty can result from trying to fit a model of limited complexity to a complex process system. Two main classes of optimisation methods are available to handle these uncertainties.

The first optimisation method is experiment-based optimisation. Instead of initially building a model, the experiment-based optimisation uses experimental measurements directly in conducting the optimisation for the print quality control. However, an experiment is different from a model prediction in the sense that it costs more time during the iterations [9]. Often it can be advantageous to use some basic model in the parametrisation of the input. This will be the approach taken in Sect. 3.4.

The second approach is robust optimisation, where it is assumed that the system is to some extent uncertain and is only known to belong to some uncertainty set. Considering a fairly accurate nominal model and a uncertainty model set, one can perform an optimisation considering the *worst-case scenario* [4, 23]. Results following this line of reasoning will be presented in Sect. 3.5.

Sections 3.4 and 3.5 treat the single-channel control problem. In Sect. 3.6 the extension towards multi-channel control is performed using the experiment-based control approach.

3.4 Experiment-Based Single-Channel Control

In this section, first an experiment-based control strategy is developed to reduce the drop velocity variations due to the residual vibrations in case that a single channel is jetting. A new input *actuation pulse* is presented, see Fig. 3.6, which consists of two trapezoidal pulses, namely, an *resonating pulse* and a *quenching pulse* [6]. This is an extension to the standard pulse given in Fig. 3.3.

The positive trapezoidal pulse (also known as the resonating pulse) is used to form and jet the drop, however this pulse cannot damp the residual vibrations generated after jetting of the ink drop. Therefore, a negative quenching pulse is added to damp the residual vibrations. The optimal pulse parameters are obtained by solving an optimisation problem, which minimises the error between the actual drop velocity and a desired drop velocity.

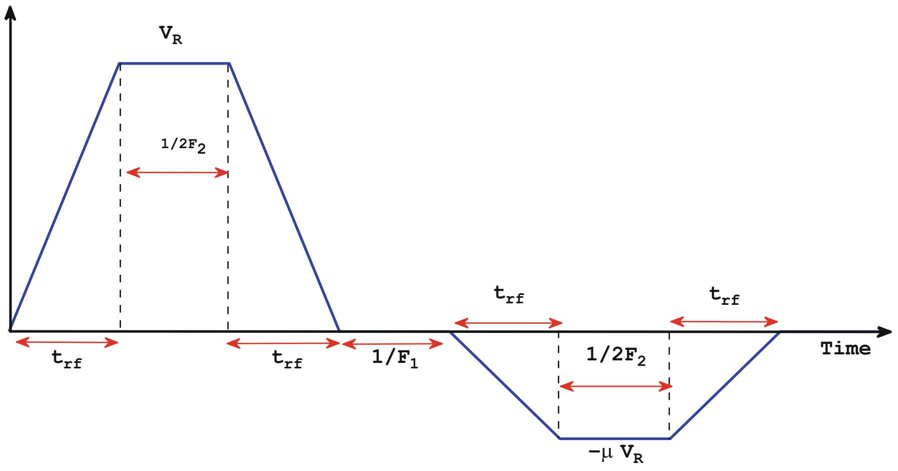


Fig. 3.6 Parameterised actuation pulse

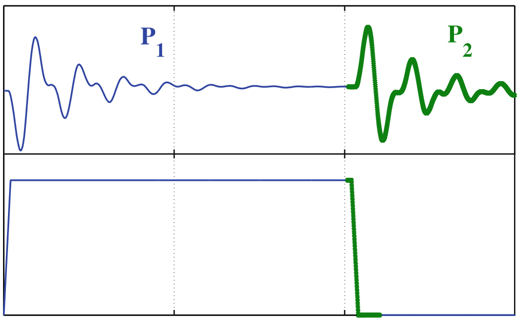


Fig. 3.7 System response to the rising and falling edge of the input pulse

3.4.1 Input Pulse Parametrisation

Based on the physical insight of the effect of the input pulse parameters on the channel acoustics, the pulse is parameterised as a function of the two resonance frequencies and the damping factor of the printhead, see Fig. 3.6. The jetting mechanism is related to the pressure wave of ink inside the channel. However, the pressure wave in ink generated by a waveform voltage is difficult to measure directly.

Piezo material can not only be used to actuate but also to measure pressure. In the setup, this self-sensing signal from the piezoelectric element was used to measure the pressure wave behaviour in a single-nozzle printhead. However, this signal has a very low signal-to-noise ratio, which makes this signal not accurate. The motion of the meniscus at the nozzle plate results from a pressure wave of ink generated from piezoelectric actuation.

The rising and falling times of the resonating pulse are used to generate the pressure wave inside the channel, while the dwell time changes the relative phase of the generated pressure waves. As explained in Sect. 3.2, at the rising edge, a negative pressure wave P_1 is generated, while a positive pressure P_2 is generated at the falling edge, see Fig. 3.7. The rising and falling times of the resonating pulse are chosen to be the same to obtain the same characteristics of the pressure waves. To achieve the maximum pressure inside the channel and, therefore, the maximum drop velocity, the pressure waves, P_1 and P_2, have to be in phase. If the pressure waves generated form the rising and falling times have the same period with a half period phase shift, an amplified pressure wave will be generated as a sum of the two pressure waves P_1 and P_2.

The period of the generated pressure waves is related to the jetting resonance frequency of the printhead, which is the second mode F_2 in the frequency response shown in Fig. 3.5. Thus, the optimal dwell time of the pulse is chosen as half of the period of the second resonance frequency F_2 as shown in Fig. 3.6. The amplitude of the resonating pulse is designed to jet a drop with specific properties, drop velocity, and drop volume.

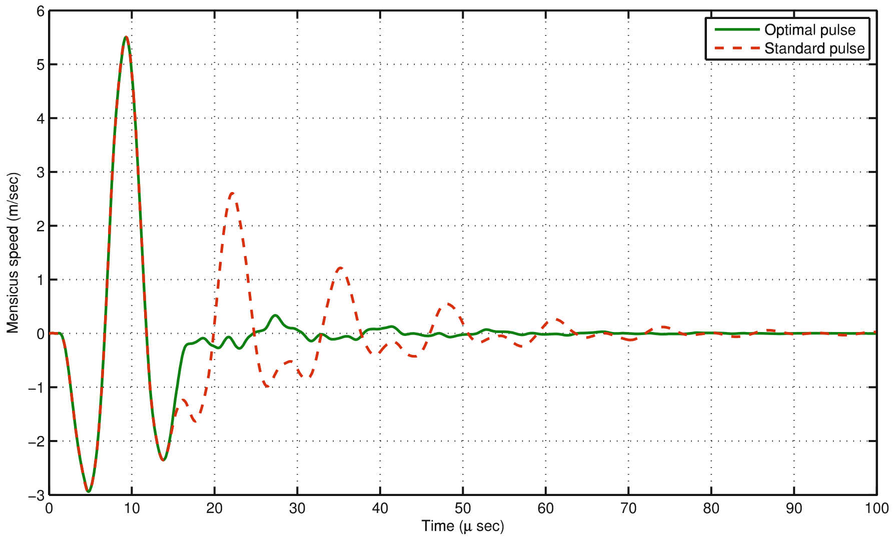


Fig. 3.8 Simulation based on the narrow-gap model

For better printing performance, the subsequent drop should not be jetted until the residual vibrations from the jetted drop have sufficiently damped out. These oscillations last for 100 μs, which limit the maximum jetting frequency to 10 kHz. The industrial applications require jetting at higher frequencies to achieve higher printing speed and higher print resolution. The quenching pulse is introduced to suppress the residual vibrations and, therefore, a higher jetting frequency is obtained. To achieve perfect cancellation of the residual oscillations, the quenching pulse is chosen similar to the resonating pulse with a negative sign. The optimal amplitude of the quenching pulse is equal to the amplitude of the resonating pulse multiplied by the damping factor μ of the printhead with opposite sign of the resonating pulse. The choice of the quenching time instant is very crucial. The quenching time should be chosen such that the pressure wave generated by the resonating pulse is anti-phase with the one generated by the quenching pulse. Therefore, the quenching pulse is placed at one period of the first resonance frequency F_1. Both the rise time t_{rf} and pulse amplitude V_R of the parametrisation shown in Fig. 3.6 are chosen to achieve a desired drop velocity.

If the exact values of the two frequency modes and the damping factor are known, the optimal pulse parameters to cope with the residual vibration can be easily obtained. To validate this new pulse parametrisation and our assumptions, the narrow-gap model is used to show that the residual vibration will be damped. We know that in this model $F_1 = 78\,\text{kHz}$, $F_2 = 160\,\text{kHz}$ and the damping factor, $\mu = 0.5$, by designing a pulse based on these parameters the residual vibration is highly damped as shown in Fig. 3.8.

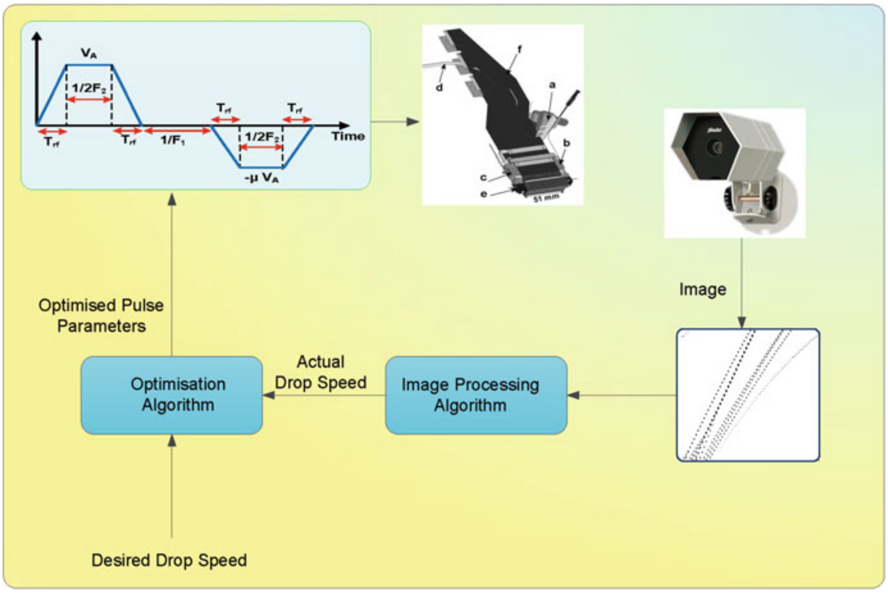


Fig. 3.9 Experiment-based optimisation feedforward control approach

3.4.2 *Setting of Experiment-Based Optimisation*

In this section, we present an optimisation-based approach to obtain the optimal parameters of the input pulse given in Fig. 3.6. In this approach, the optimisation is carried out on a real setup instead of using a printhead model, hence all modelling issues are circumvented. A schematic diagram of the approach is illustrated in Fig. 3.9. We are concerned with improving the drop properties, mainly the drop velocity, generated by the printhead shown central in the upper part of the figure. Therefore, we optimise the input pulse based directly on the drop velocity measured with a camera (upper right). This is different from the approach in Sect. 3.3 where the meniscus velocity is optimised. In this approach, a high-speed camera is used to capture the drop, which is travelling from the nozzle plate to the substrate. A time history of the jetted drop is obtained. Using an image processing technique, the velocity of jetted drops is computed. The input pulse is optimised such that the error between the measured drop velocity and a desired velocity is minimised. The optimisation process is done with a real printhead in the loop.

3.4.3 *Image Processing*

Image processing refers to the use of computer algorithms performed on a digital image. The purpose is to transform a digital image into another digital image which

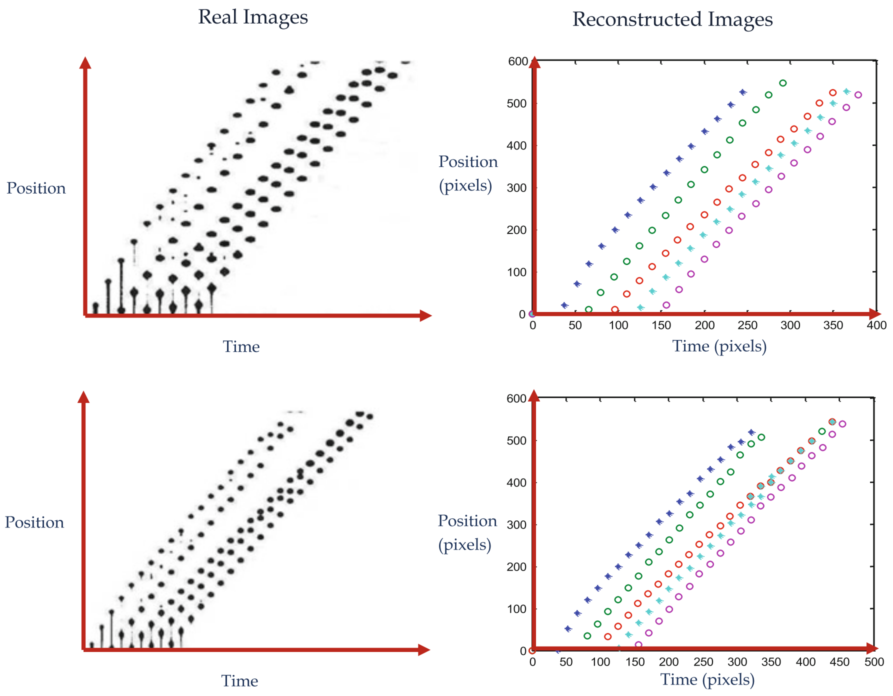


Fig. 3.10 Samples of the output of the image processing algorithm. Position is the distance to the nozzle plate

is usually used for image coding, image enhancement, image restoration, and/or image feature extraction [11, 29]. In our approach, an image processing technique is used for feature extraction. The goal of the image feature extraction technique is to transform the image into another image from which the required measurement can be derived.

Image-based measurement has been widely applied in various kinds of scientific applications as well as many industrial and medical applications. Vision feedback control has been introduced in order to increase the flexibility and the accuracy of robotic systems [13]. The aim of the visual servo approach is to control a system using the information provided by a vision system. It involves visual tasks, which are used in real-time control.

In our approach, a high-speed camera is used to record the time history of the drops travelling from the nozzle plate to the print media. An image processing technique is developed to retrieve the actual velocity of each drop. A sample of the time history of 5 drops is shown in Fig. 3.10. As depicted, these images include both the jetted drops and small satellite drops. The image processing algorithm should be capable to extract only the information of the jetted drops. Towards this objective, the image is converted into a binary image, which has only two possible values for each pixel. Typically the two colours used for a binary image are black and white, where a “0” is assigned for black and a “1” for white. After that, the image is filtered to remove the small dots that represent the satellites. The filter used in our approach, creates a flat disk-shaped structuring element with a specific neighbourhood. The neighbourhood is defined as a matrix containing 1’s and 0’s; the location of the 1’s defines the neighbourhood for the morphological operation. The centre of the neighbourhood is its centre element. Based on the number of 1’s in the neighbourhood, the image is filtered. Finally, a pattern recognition technique based on the 2D pixel search is developed to obtain the position of each drop. Figure 3.10 shows two examples of the time history of 5 drops and the reconstructed image. Once the image is reconstructed, the velocity of each drop is computed based on the drop position and the travel time.

3.4.4 Optimisation Problem

Experiment-based optimisation adopts experimental measurements as function evaluations for optimisation. Iterative points are generated by a proper algorithm that provides the direction of improvement for the optimisation problem. A superior optimisation algorithm, which reduces the number of test experiments, is desirable.

As explained in Sect. 3.4.1, the input pulse, see Fig. 3.6, is parameterised as,

$$\theta = [t_{\mathrm{rf}} \ \frac{1}{2F_2} \ t_{\mathrm{rf}} \ V_R \ \frac{1}{F_1} \ t_{\mathrm{rf}} \ \frac{1}{2F_2} \ t_{\mathrm{rf}} \ -\mu V_R],$$

where t_{rf} denotes the rise and fall time, V_R the resonating pulse amplitude, F_1 and F_2 the first and the second resonance frequency of the printhead, respectively, and μ the damping factor of the channel.

The optimisation problem is defined as the minimisation of the cost function

$$\mathscr{J}(\theta) = \sum_{f=F_{\min}}^{F_{\max}} \sum_{t=0}^{T} (v_{\mathrm{desired}} - v_{\mathrm{actual}}(\theta, t, f))^2, \tag{3.5}$$

subject to

$$\theta_{\min} \leq \theta \leq \theta_{\max}, \tag{3.6}$$

where f is the jetting frequency, which is the basic frequency of jetting a train of drops, v_{desired} is the desired drop velocity, v_{actual} is the actual drop velocity, T denotes the total time of the experiment, and t is the time instant when a measurement is taken. The optimal pulse that minimises the cost function is given by

$$\theta_{\mathrm{opt}} = \arg\min_{\theta} \mathscr{J}(\theta), \tag{3.7}$$

where the optimisation variable θ is defined as

$$\theta := [t_{\text{rf}}\ F_1\ F_2\ \mu\ V_R].$$

This problem formulation leads to a non-linear optimisation problem. A standard optimisation algorithm is used to solve this constrained non-linear optimisation problem, which is not convex. The search algorithm can be generally categorised into two types, gradient-based and gradient-free methods [1, 22]. Gradient-based algorithms utilise gradients to provide search directions for improvement. The calculation of the gradient at a given point can be conducted by perturbations. Finite differencing is one method widely used for gradient calculations. Generally, gradient-based algorithms converge faster as the gradient leads to the right search direction for minimisation. The gradient calculation based on the finite differencing for the experiment-based optimisation, however, needs a large number of experiments in case of a large number of variables. On the other hand, the gradient-free algorithm can be performed without computing the gradients, which in turn reduces the number of experiments required for the optimisation. One of the standard gradient-free methods is based on Nelder-Mead simplex [21]. However, this technique is a heuristic search method that can converge to non-stationary points. Therefore, we used a standard gradient-based algorithm to solve the optimisation problem (3.5), namely the trust region reflective algorithm [26]. This algorithm can efficiently handle the non-linear constrained optimisation problem. To lower the risk of being trapped in a local minimum, the optimisation problem is carried out several times using different initial values.

3.4.5 *Experimental Results*

In this section, the optimised pulse is applied to a real printhead and the results are compared to a standard pulse. Several tests have been carried out to evaluate the efficiency of the optimised pulse.

3.4.5.1 Continuous Jetting

Figure 3.11 shows the DoD drop velocity curve for the optimised pulse and the standard pulse. The DoD curve is the velocity of the drop at several jetting frequencies when the nozzle is continuously jetting. The drop velocity variation for the optimised pulse is less than 1.4 m/s compared to 6 m/s in case of the standard pulse.

We observe that the main contribution of the error is due to the low drop velocity at jetting frequencies from 20 to 30 kHz. The reason for the slow drop velocity at this frequency range may be that the two frequency modes, F_1 and F_2, and damping factor μ of the printhead are different at this frequency range. Therefore, we focus on optimising a second pulse only over this frequency range, 20 to 30 kHz, and we obtained a new optimised pulse for this range.

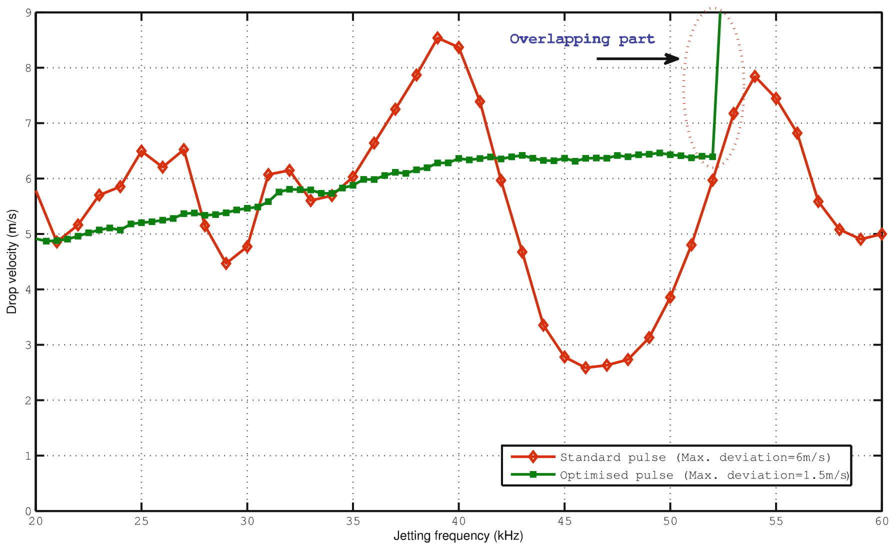


Fig. 3.11 DoD curve comparison of the standard and optimised pulses

Now by defining two pulses, one for the low jetting frequencies, 20 to 30 kHz, and another one for the rest of the jetting frequencies, the maximum drop velocity variation is less than 0.9 m/s as shown in Fig. 3.12. As depicted in Fig. 3.13, the optimised pulse at the low-frequency range is not just a scaled version of the high-frequency optimised pulse. Note that the optimised pulses can be used for jetting drops with a DoD frequency up to jetting frequency 56 kHz without overlapping of the pulse. The sudden change in the DoD velocity curve at the jetting frequency of 56 kHz is due to hardware limitations since the waveform generator cannot overlap pulses.

3.4.5.2 Jetting a Train of Drops

The second test is jetting a train of drops at several jetting frequencies and analysing the time history of the drop travelling from the nozzle plate to the print media. Figure 3.14 shows the time history of the train of drops for both the experiment-based optimised pulse and standard pulse, with a jetting frequency of 48 kHz. Figure 3.14 compares the experimental results of using the optimised pulse (a) with the standard pulse (b).

The application of the optimised pulse results in all 16 drops travelling at the same velocity. The drops are disposed at an equal distance on the print media. The first drop is, however, slower and therefore the drop is merged with the second drop. A small satellite drop is visible after the last drop. However, jetting with the standard pulse results in drops where the first drop travels to the print media with

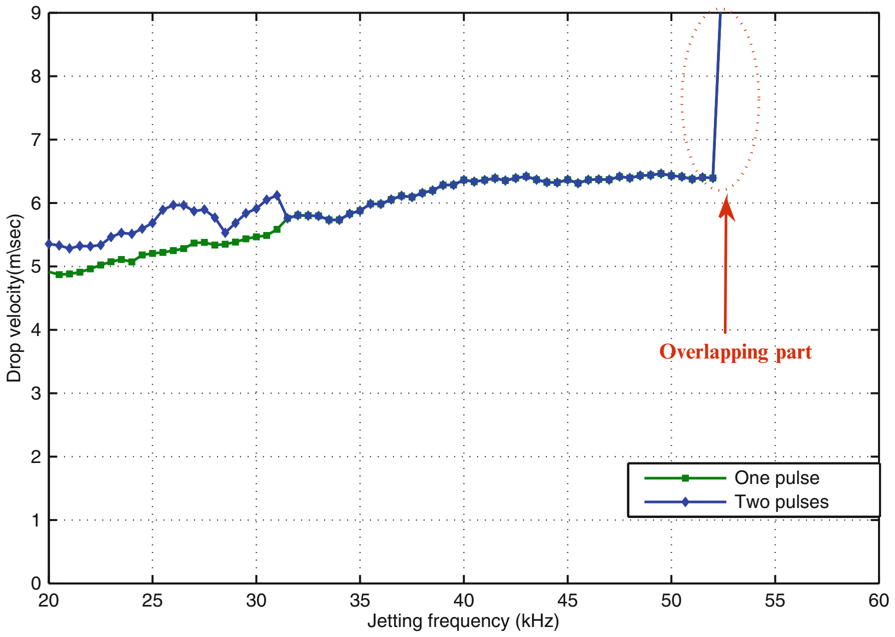


Fig. 3.12 Optimised DoD curve with two optimised pulses

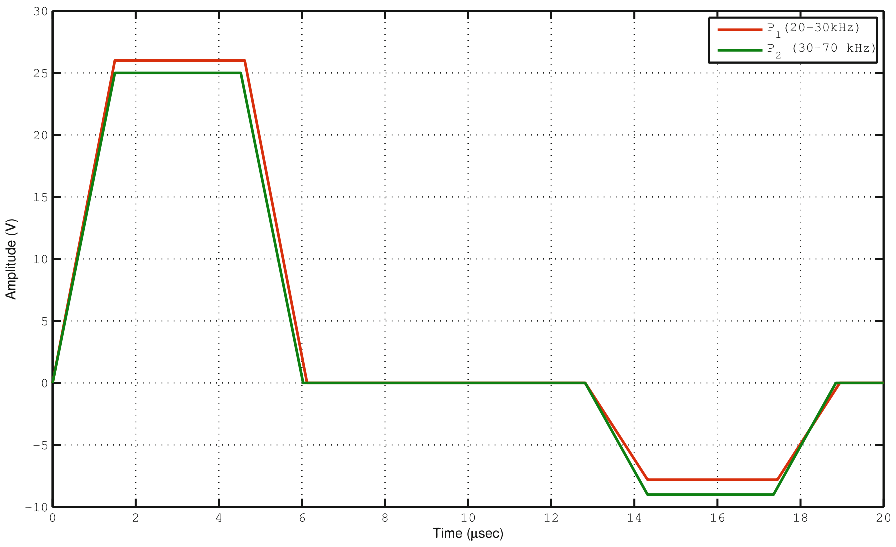


Fig. 3.13 Optimised pulses based on vision-based optimisation

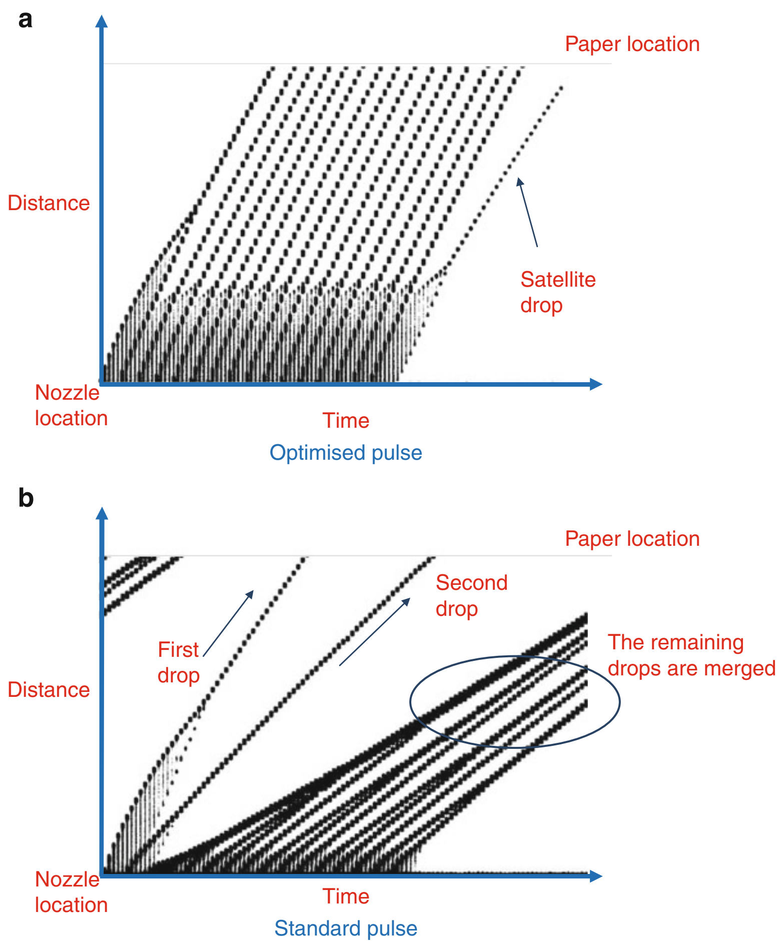


Fig. 3.14 Jetting 16 drops at a DoD frequency of 48 kHz

a high velocity, the subsequent drops travel with different velocities. Several drops are merged into a single large drop. Consequently, the drops are misplaced on the print media and have different sizes.

Several experiments have been carried out for various jetting frequencies ranging from 20 to 70 kHz and the performance of the printhead has been analysed in terms of the drop velocity. The drop velocity of jetting 10 drops over jetting frequencies 20 to 70 kHz is depicted in Figs. 3.15 and 3.16, for the standard pulse and the vision-based feedforward optimised pulse, respectively. The vision-based feedforward pulse shows less drop velocity variation than the standard pulse.

The performance is evaluated based on the maximum drop velocity variation over the whole range of the DoD frequencies, the behaviour of the first drop, and the maximum drop velocity variation at each DoD frequency. For the standard pulse,

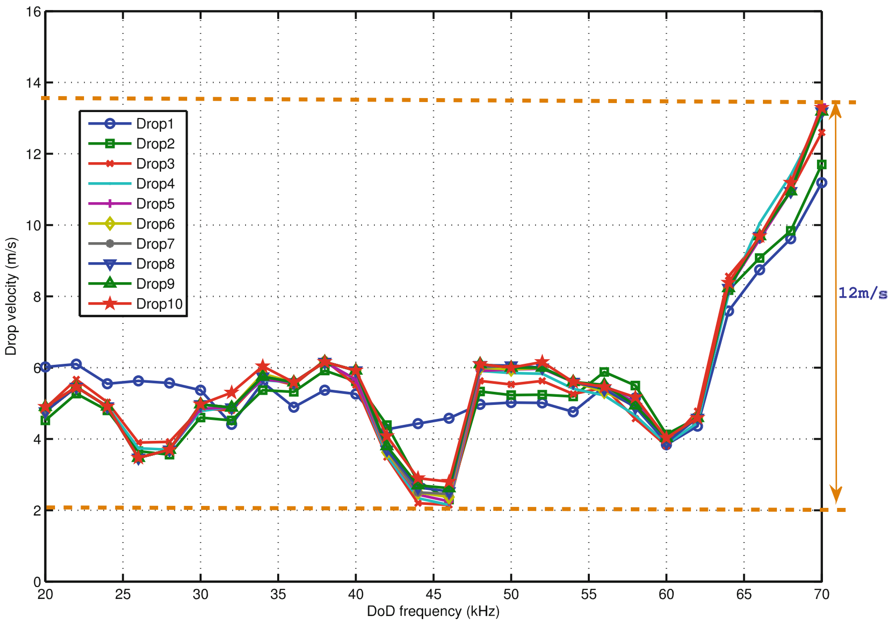


Fig. 3.15 Jetting 10 drops at different DoD frequencies, 20 to 70 kHz using the standard pulse

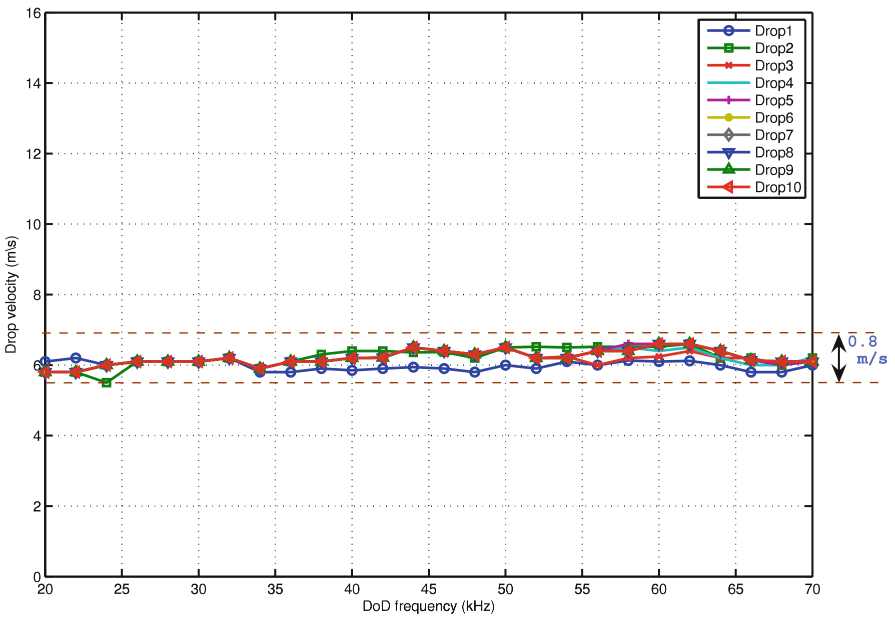


Fig. 3.16 Jetting 10 drops at different DoD frequencies using the two optimised pulses

the drop velocity varies from 2 to 13.5 m/s, which is a considerable variation over the whole DoD frequency range. Figure 3.15 shows that the first drop behaves in a completely different manner than the subsequent drops. At 65% of the frequencies, the first drop is faster than the remaining drops. As a consequence, a poor print quality is obtained and a shadow appears in the printed bitmap. Moreover, the maximum drop velocity variation at each jetting frequency is around 3 m/s, which is computed as

$$\Delta v(f) = v_{\max}(f) - v_{\min}(f), \tag{3.8}$$

and

$$\Delta v_{\max} = \max_{f} \Delta v(f). \tag{3.9}$$

3.4.5.3 Summarising the Results

A large improvement has been achieved by implementing the vision-based feedforward pulse. The drop velocity variation, over the considered frequency range, is reduced to 0.8 m/s compared to 12 m/s for the standard pulse. Figure 3.16 shows that the difference between the first drop the subsequent drops is negligible. Therefore, a bitmap printed with the optimised pulse does not contain the shadow effect. At each DoD frequency, the figure shows that all the drops have a much more similar velocity. The maximum drop velocity variation defined in (3.9) is less than 0.6 m/s for the optimised pulse compared to 3 m/s for the standard pulse. The requirement of ± 0.25 m/s is thereby almost achieved for the full adaptive setting.

Summarising, the experiment-based optimisation scheme is proposed to minimise the drop velocity variations. With this scheme there is no need for an accurate model. The pulse design is based on controlling the drop velocity and not on an intermediate state. Further, a new pulse parametrisation was proposed. The input pulse is parameterised as a function of the frequency response of the printhead that leads to fewer optimisation parameters. Owing to this parametrisation on-line adaptation of the pulse is possible. If the pressure inside the channel (or meniscus velocity) is measured on-line, the damping factor μ and the two frequency modes F_1 and F_2 are obtained by computing the frequency content of the measured variable.

Experiment-based optimisation is able to deal with uncertainty in the process. The alternative of robust optimisation will be discussed in the next section.

3.5 Model-Based Single-Channel Control

In the previous section, it has been shown that the printhead installed in the experimental setup can be used to design actuation pulses. However, once the printhead is installed in a document printing system it is not possible to measure the drop

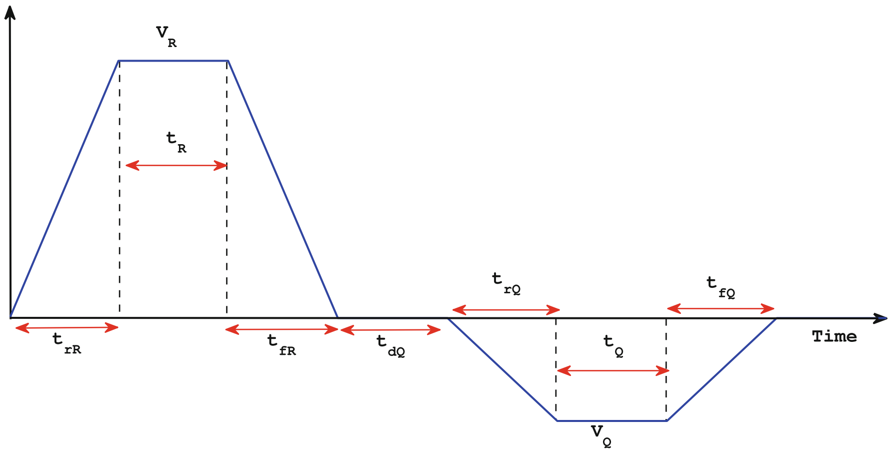


Fig. 3.17 Proposed actuation pulse

velocity. This is due to the fact that providing instruments inside a document printer to measure the drop velocity would lead to a complex and expensive system. In such a scenario, a fairly accurate model of the printhead can be used to (re-)design actuation pulses. As the printhead characteristics can change over time, this provides means to adapt the input in order to maintain a high performance level. In this section, first a method is presented which uses only the nominal model $H(q)$ to design the actuation pulse for a single ink channel. This approach is further extended to the design of a robust actuation pulse which can handle model uncertainties.

3.5.1 Optimisation-Based Feedforward Control Design

In order to design the piezo actuation pulse, the model $H(q)$ introduced in Sect. 3.3 is used. This model describes the dynamics from the piezo input $u(k)$ to the meniscus velocity $y(k)$. We have parameterised the set $u(k,\theta)$ of actuation pulses that the driving electronics can generate see Fig. 3.17.

The parameters that are optimised are the rise time (t_r), the dwell time (t_w), the fall time (t_f) and the amplitude (V) of both the resonating (R) and the quenching pulse (Q). The time interval between the resonating pulse and the quenching pulse is t_{d_Q}. Thus, an actuation pulse $u(k,\theta)$ is defined by the parameter vector

$$\theta = [t_{r_R}\ t_{w_R}\ t_{f_R}\ V_R\ t_{d_Q}\ t_{r_Q}\ t_{w_Q}\ t_{f_Q}\ V_Q]^T.$$

Note that the amplitude of the pulse is expressed in Volts. The time parameters in θ are the number of samples. However, for simplicity, we express these parameters in μs by multiplying the number of samples with the sampling time T_s. Now, we

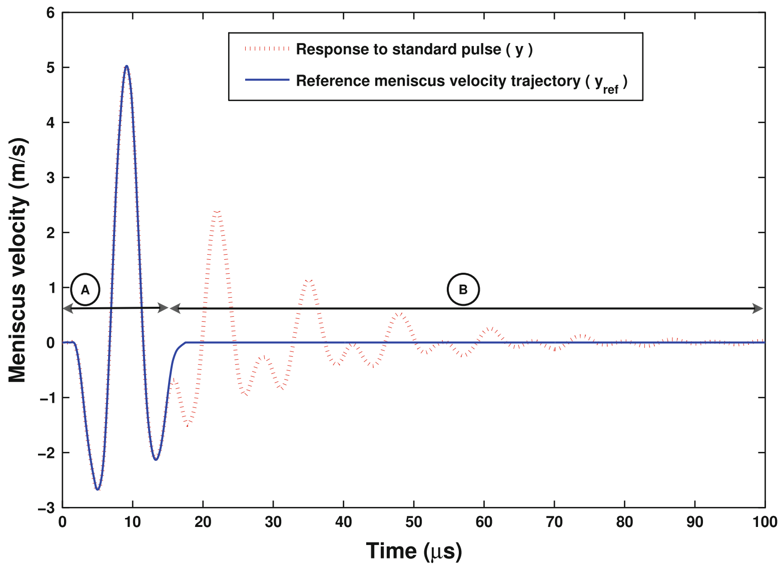


Fig. 3.18 Reference meniscus velocity trajectory

design a template $y_{\text{ref}}(k)$ for the desired meniscus velocity so that a meniscus velocity profile with fast-decaying residual oscillations is obtained. Based on this template $y_{\text{ref}}(k)$ and the transfer function $H(q)$ an optimal actuation pulse $u(k, \theta_{\text{opt}})$ is computed by minimising the squared tracking error $e^2(k) = (y_{\text{ref}}(k) - y(k))^2$.

3.5.1.1 Design of the Reference Meniscus Velocity

To design the template $y_{\text{ref}}(k)$, we determine the response of $H(q)$ when the standard pulse $u_{\text{std}}(k, \theta)$ is used as an input as earlier shown in Fig. 3.3. The standard pulse is known to jet a drop with the desired properties when the ink channel is at rest. The response $y(k)$ of $H(q)$ to this pulse has two parts as shown in Fig. 3.18 (dashed line). Part A of the response allows the drop to be jetted at the desired drop velocity: the template $y_{\text{ref}}(k)$ for the meniscus velocity should therefore be the same as $y(k)$ in Part A. Part B of the response $y(k)$ represents the residual oscillations. This is an undesired behaviour since the residual oscillations perturb the subsequent drops. Therefore, in Part B, we force the desired meniscus velocity $y_{\text{ref}}(k)$ to zero.

In Sect. 3.3, we have seen that if the ink channel is actuated with similar initial conditions then the resulting drops will have similar properties. If the actuation pulse is designed in such a way that the meniscus velocity $y(k)$ follows the reference trajectory $y_{\text{ref}}(k)$, then the channel will come to rest very quickly after jetting the ink drop. Thus, it will create the conditions to jet the ink drops at higher DoD frequencies.

3.5.1.2 Optimal Actuation Pulse Design

In [16], we define the optimal actuation pulse as the trapezoidal input $u(k,\theta)$ which minimises the difference between the reference trajectory $y_{\text{ref}}(k)$ and the meniscus velocity $y(k) = H(q)u(k,\theta)$. More precisely, the parameter vector θ_{opt} describing the optimal pulse is the one solving the following (non-linear) optimisation problem:

$$\theta_{\text{opt}} = \arg\min_{\theta} \mathscr{J}_{\text{nom}}(\theta), \quad \text{subject to} \quad \theta_{\text{LB}} \le \theta \le \theta_{\text{UB}}, \tag{3.10}$$

with

$$\mathscr{J}_{\text{nom}}(\theta) = \sum_{k=0}^{N} \Big(y_{\text{ref}}(k) - H(q)u(k,\theta) \Big)^2$$

where $N = \frac{T}{T_s}$, T_s is the sampling time, T is chosen equal to 100 µs, q is the forward shift operator, and θ_{LB} and θ_{UB} are vectors containing the lower and the upper bounds on each element of the parameter vector θ.

The optimal actuation pulse $u_{\text{opt}}(k) = u(k,\theta_{\text{opt}})$ is shown in Fig. 3.23. Simulation results show that the optimal pulse $u_{\text{opt}}(k)$ tracks the reference $y_{\text{ref}}(k)$ and damps the residual oscillations almost perfectly (see [16]). Since the model is not perfect, results should be verified experimentally.

3.5.1.3 Experimental Results with the Optimal Pulse

To validate this approach, the performance of the optimal pulse $u_{\text{opt}}(k)$ has been compared to the performance of the standard pulse $u_{\text{std}}(k)$ in an experimental setup. The experimental test consists of jetting 10 ink drops from the inkjet channel at a fixed DoD frequency. This experiment was repeated for different DoD frequencies ranging from 20 to 70 kHz.

The drop velocities of the 10 drops are shown in Figs. 3.19 and 3.20 as a function of the DoD frequency (DoD curve). Figure 3.19 is obtained when the standard pulse $u_{\text{std}}(k)$ is used and Fig. 3.20 shows the results when the optimal pulse $u_{\text{opt}}(k)$ is used. From these figures, it is obvious that the optimal pulse leads to a DoD curve which is much flatter. For the standard pulse, it can be seen that the maximum drop velocity variation over the DoD frequency range is 12 m/s while with $u_{\text{opt}}(k)$ this variation is reduced to 4.5 m/s. Currently, commercial printers, which use the standard pulse, print at a fixed speed to compensate for very large drop velocity variations. Note that our goal is to achieve variable printing speed, hence reduction in the drop velocity variation over the DoD frequency range with the optimal pulse is quite significant. Even when printing at a fixed speed with the optimal pulse, as the DoD curve is relatively flat, the robustness of the printing system is improved.

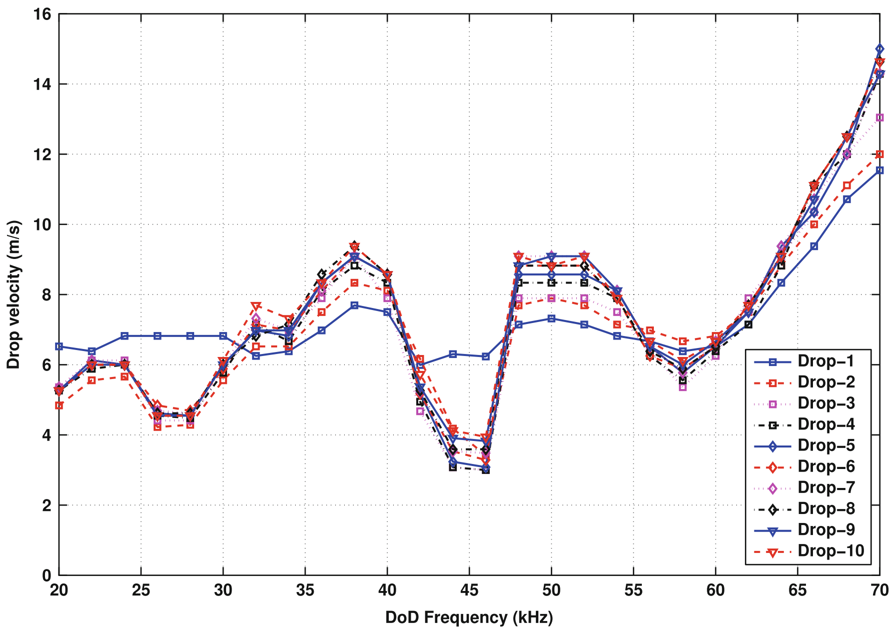


Fig. 3.19 Experimental DoD curve with the standard pulse $u_{\text{std}}(k)$

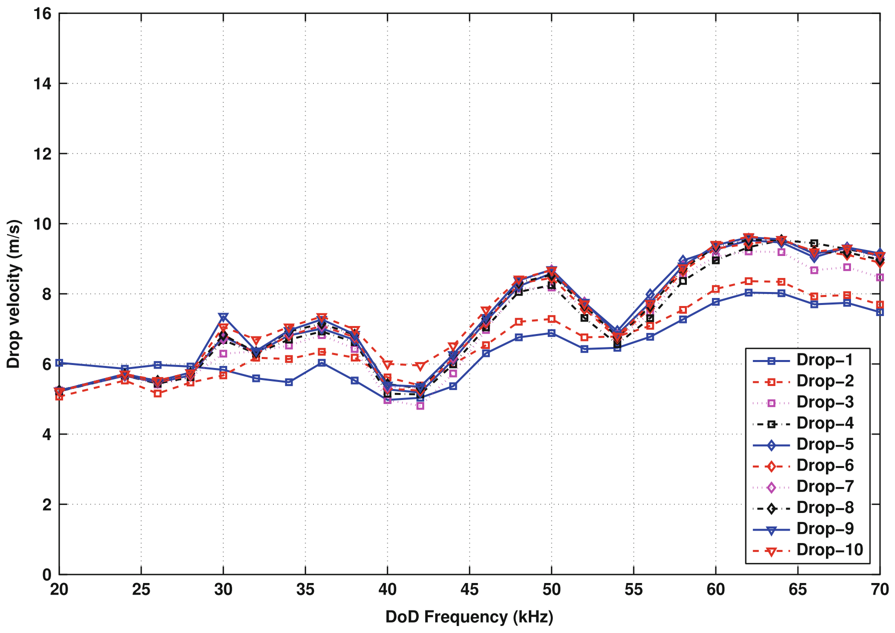


Fig. 3.20 Experimental DoD curve with the optimal actuation pulse $u_{\text{opt}}(k)$

3.5.2 Robust Feedforward Control Design

The experimental results, presented in the previous section show that the optimal actuation pulse $u(k, \theta_{\mathrm{opt}})$ improves the performance of the inkjet printhead compared to the standard actuation pulse (the pulse that is generally used for the printhead). However, the DoD curve with the optimal pulse $u(k, \theta_{\mathrm{opt}})$ is not completely flat. The main reason is that the model $H(q)$ used for the design of the optimal pulse only represents the dynamics of the system when a single drop is jetted. The model $H(q)$ is therefore not representative for the dynamics of the system when jetting a series of drops at a certain DoD frequency. Data-based identification of the system dynamics furthermore suggests that the dynamics from the piezo input to the meniscus velocity depend on the DoD frequency [15]. In this section, we discuss reasons for this DoD dependency and further, we present an uncertain system model representation to cover the set of models at all DoD frequencies. Subsequently, the robust actuation pulse is obtained by minimising the worst-case norm of the tracking error with the uncertain model.

3.5.2.1 Uncertain Inkjet System

As discussed in the introduction, at different DoD frequencies, the dynamics from the piezo input to the meniscus velocity $H(q)$ will not be the same. One possible reason for this frequency-dependent behaviour is the fact that the dynamics of the refilling of the ink channel after a drop has been jetted is not modelled in the narrow-gap model which is the basis for $H(q)$. By jetting drops at different DoD frequencies, these slow refill dynamics may change the average meniscus position at the time of drop separation. This may in turn change the properties of the linear model $H(q)$ when obtained at different DoD frequencies. The other reason may be non-linear effects in the drop formation process. In order to investigate this phenomenon, we have used experimental identification. It is very difficult to experimentally measure the meniscus position and the meniscus velocity while jetting an ink drop. However, the piezoelectric crystal can be simultaneously used as an actuator and as a sensor. Note that this sensor signal is reconstructed using an off-line method which requires two experiments to be done, one with ink inside the channel and other by removing ink from the channel (for details, see [14]). Therefore, we have identified a transfer function from the piezo input to the piezo-sensor output (which is proportional to the derivative of the ink channel pressure) for a given DoD frequency. We have done several such experiments at various fixed DoD frequencies in the operating range of the inkjet printhead. The details of the identification experiments are omitted (for details, see [15]). Readers interested in the identification methods are recommended to read [19].

We propose to represent the set of dynamic models at different DoD frequencies by a nominal system with a compact uncertainty on its parameters. In order to reduce the complexity, we select a simplified discrete-time model $H_{\mathrm{low}}(q)$ based on $H(q)$.

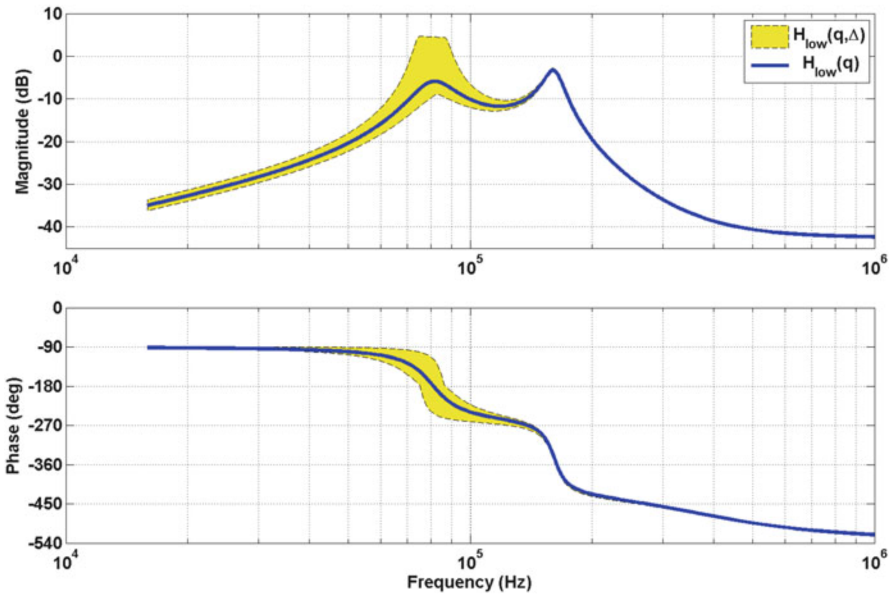


Fig. 3.21 Frequency response of the transfer function model $H(q)$

We know that higher-order modes in the meniscus velocity do not contribute significantly to the drop formation process [5]. Hence, these higher-order modes are neglected in $H_{\text{low}}(q)$. The discrete-time model $H_{\text{low}}(q)$ describes the dynamics from the piezo input voltage u to the meniscus velocity y. The transfer function $H(q)$, parameterised by coefficients g, a_i, and b_i, is given as follows

$$H_{\text{low}}(q) = g\left(\frac{q^2+b_1q+b_2}{q^2+a_1q+a_2}\right)\left(\frac{q^2+b_3q+b_4}{q^2+a_3q+a_4}\right), \tag{3.11}$$

where q is the forward shift operator. These two second-order systems can describe two resonant modes. Figure 3.21 shows the frequency response of this fourth-order transfer function $H(q)$ as a solid line.

It is observed that the first resonant mode of the inkjet system varies a lot with the change in the DoD frequency. Using identification results [15] and some physical insight, it is assumed that for the first resonant mode of $H(q)$, the resonance frequency ω_{n1} variation is approximately in the interval $[-7\ +7\%]$ and the damping ζ_{n1} variation is approximately in the interval $[-70\ +30\%]$. The variation in the second resonant mode is relatively small compared to the first one and we neglect it in order to obtain a simpler and more compact uncertainty description. This is a valid assumption since the first mode greatly influences the ink drop properties [5] compared to the second mode. The details of the relation between the uncertainty on the properties of the resonant mode and the coefficients of the transfer function

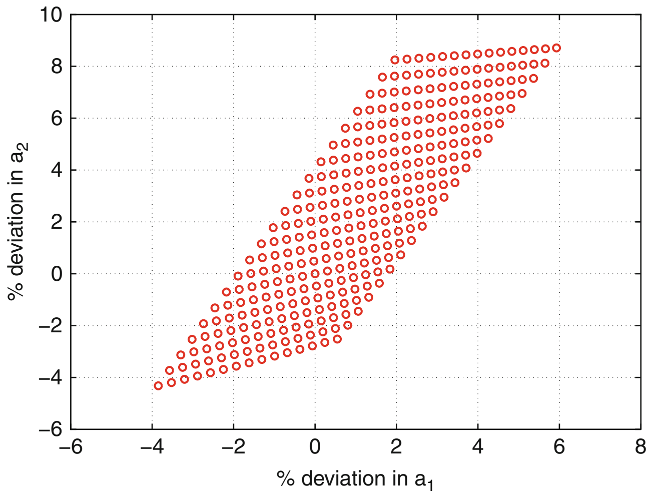


Fig. 3.22 Parametric uncertainty relative to the nominal value

(3.11) can be found in [17]. Due to the uncertain first resonant mode, the coefficients of the transfer function (3.11), a_1 and a_2 are subjected to uncertainty Δ. The uncertainty $\Delta = [\Delta^{(1)}\ \Delta^{(2)}]^T$ perturbs the coefficients a_1 and a_2 in the following manner:

$$a_1(\Delta) = a_{1,\text{nom}}(1 + \Delta^{(1)}) \tag{3.12}$$

$$a_2(\Delta) = a_{2,\text{nom}}(1 + \Delta^{(2)}), \tag{3.13}$$

where $a_{1,\text{nom}}$ and $a_{2,\text{nom}}$ are the nominal values of the coefficients a_1 and a_2. This means that the uncertainty Δ on the coefficients a_1 and a_2 lie in the set $\mathfrak{D}$ given in Fig. 3.22. Note that the variations in a_1 and a_2 are coupled unlike the variations in ω_{n1} and ζ_{n1} due to the continuous-time domain to discrete-time domain mapping (see [17]).

Now, the set of dynamic models obtained at various operating DoD frequencies can be represented by the uncertain inkjet system $H(q,\Delta), \Delta \in \mathfrak{D}$. The frequency response of the uncertain inkjet system which is represented by $H(q,\Delta)$ is shown by the shaded area in Fig. 3.21. Note that even though the inkjet system is described by an abstract model (3.11), the experimental observation of DoD dependency can be incorporated in this model. The details about different type of uncertainties and ways to incorporate them in a dynamic model are given in [31].

3.5.2.2 Robust Actuation Pulse Design

As discussed in the previous section, we have the set of multiple models represented by the uncertain inkjet system $H(q,\Delta)$ which is perturbed by the uncertainty $\Delta \in \mathfrak{D}$. Therefore, we should design a robust actuation pulse which ensures a minimum worst-case performance over the polytopic uncertainty $\mathfrak{D}$, rather than obtaining an optimal actuation pulse whose performance is only good for the nominal inkjet system $H_{\text{low}}(q)$. As the dimension of the parameter space of Δ is only 2, we can easily grid the parametric uncertainty $\mathfrak{D}$. Let the set $\mathscr{S}$ be the grid on the parametric uncertainty $\mathfrak{D}$, defined as

$$\mathscr{S} = \{\Delta^i, i = 1,\ldots,m, \mid \Delta^i \in \mathfrak{D}\}. \tag{3.14}$$

For a given parameter vector θ, we define the robust performance index $\mathscr{J}_{\text{robust}}(\theta)$ as the worst-case sum of squared error computed at each of the m grid elements, i.e.

$$\mathscr{J}_{\text{robust}}(\theta) = \max_{\Delta^i \in \mathscr{S}} \sum_{k=0}^{N} \Big(y_{\text{ref}}(k) - H_{\text{low}}(q,\Delta^i)u(k,\theta) \Big)^2. \tag{3.15}$$

Now, the constrained robust actuation pulse parameter is thus the solution θ_{robust} of the following optimisation problem

$$\min_{\theta} \mathscr{J}_{\text{robust}}(\theta), \quad \text{subject to} \quad \theta_{\text{LB}} \leq \theta \leq \theta_{\text{UB}}, \tag{3.16}$$

where, θ_{LB} and θ_{UB} are the vectors containing the lower and the upper bounds on each element of the parameter vector θ.

This is a non-linear optimisation problem and can be solved off-line using standard optimisation algorithms similar to problem (3.10).

3.5.2.3 Experimental Results with the Robust Pulse

In this section, we present experimental results to show the improvements in the drop consistency with the robust actuation pulse. The robust actuation pulse parameter vector θ_{robust} obtained after solving the optimisation problem using the `fmincon` command of MATLAB [20]. The robust pulse $u_{\text{robust}}(k) = u(k,\theta_{\text{robust}})$ designed in such a manner is shown in Fig. 3.23 as a solid line.

We have done similar experiments with the robust pulse $u_{\text{robust}}(k)$ and obtained the DoD curve shown in Fig. 3.24. Since the robust pulse $u_{\text{robust}}(k)$ is designed to give a good average performance over the set of models represented by $H(q,\Delta)$, it delivers better results than the optimal pulse. Now, the drop velocity variation is less than 2 m/s.

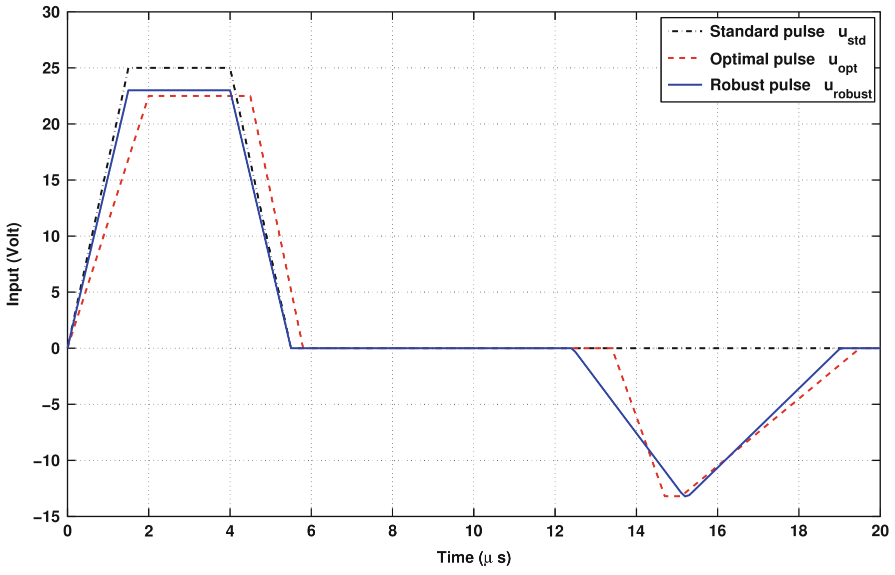


Fig. 3.23 Actuation pulses

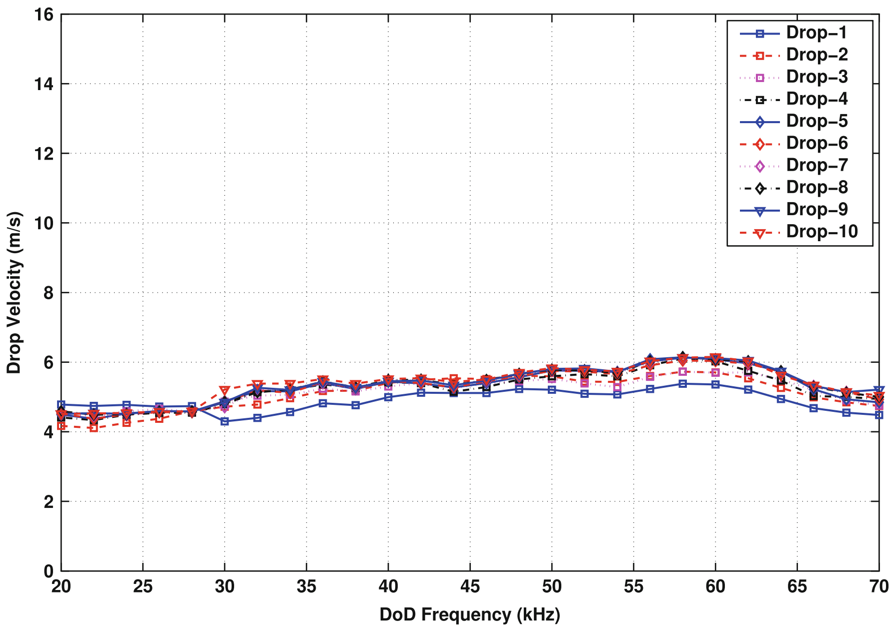


Fig. 3.24 Experimental DoD curve with the robust pulse $u_{\text{robust}}(k)$

The overall improvement in the velocity consistency achieved using the optimal pulse and the robust pulse has far-reaching consequences for the print quality. This is because the distance between the inkjet printhead and the print paper is relatively large considering the higher drop velocity.

Summarising, it is evident that the model-based control approach is an effective way to design the input waveform for an inkjet system. When the system dynamics are represented by a model with good accuracy, then the optimal pulse is obtained as the one minimising the tracking error. For an inkjet printhead, we have seen that the set of dynamic models obtained at different DoD frequencies can be represented by a compact parametric uncertainty $\Delta \in \mathfrak{D}$ on the nominal model of the inkjet system, i.e. $H_{\text{low}}(q,\Delta)$. The robust pulse, which is obtained as the solution of the proposed optimisation problem, ensures a minimum performance for all models in the uncertainty set $H_{\text{low}}(q,\Delta)$. Experimental results have shown that this approach delivers a fairly flat DoD curve, which is desired for variable-speed printing. In the future, we will extend this approach for the MIMO case to tackle the crosstalk problem.

3.6 Experiment-Based Multi-channel Control

With multi-channel control one has to deal with the crosstalk effect next to the residual vibrations. Therefore the ink channels are divided into a number of groups and a time delay between the actuation of the channels is introduced. An optimisation problem is formulated to compute the optimal time delay. In the next step, we explain the fact that the drop properties depend on the bitmap to be printed. Therefore, based on the bitmap the input pulse is updated. The 0-pixel in the pattern will result in succeeding drops with different properties. The input pulse has to be updated based on the number of preceding 0-pixels in the pattern. Thus an optimisation problem is formulated for several jetting patterns. This optimisation problem results in a set of pulses, which reduces the drop velocity variations for any bitmap.

As inkjet channels become larger, the crosstalk between closely-spaced firing channels becomes more severe resulting in an adverse impact on print quality. A method of addressing this problem, which is applicable to the DoD inkjet without any physical changes to the ink channel design and with minimal changes to the channel configuration, is described. The individual channels are divided into N interspersed groups and the permitted firing time of each group has its own small time delay T_D. The time delay is optimised for each group such that when all channels are fired, a maximum amount of crosstalk cancellation is achieved. Because the firing channels are the source of the acoustic wave train, predominantly at a resonance frequency of the channel, the interaction with neighbouring firing channels depends strongly on the phase relationship with the arriving wave. For a given printhead, several basic experimental measurements are carried out, and the

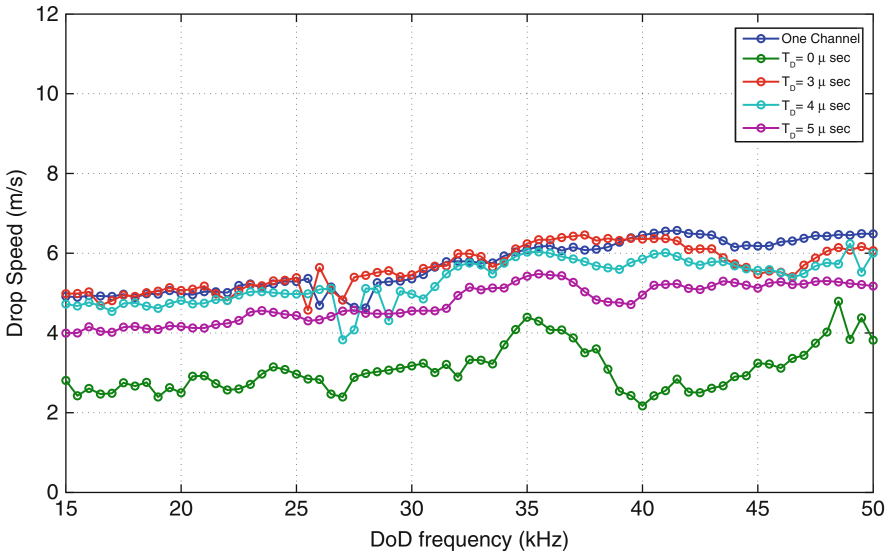


Fig. 3.25 The DoD velocity curve at different time delays between the odd and even channels

data are used to calculate the change in drop velocity as a function of the delays for a channel near the centre of the array when all channels are firing.

The piezoelectric actuators are divided into groups and time delay is applied between the inputs of these groups. For example, all the odd channels are fired and then after the time delay the even channels are fired. This delay scheme is known as a two-phase firing scheme. Another alternative is the three-phase scheme, where one third of the channels are fired simultaneously, channels $1,4,7,\ldots$, and then after a time delay T_{d1} the second third of the channels are fired, channels $2,5,8,\ldots$. Finally, the third group of the channels, channels $3,6,9,\ldots$, are fired after time delay T_{d2}. Similar delay schemes, such as four-phase or five-phase, can be also implemented.

3.6.1 *Minimisation of the Crosstalk*

In this section, we consider the two-phase firing scheme to minimise the effect of the crosstalk between the ink channels. Figure 3.25 illustrates the effect of crosstalk on the drop velocity. This figure shows the DoD velocity curve for one jetting channel and the DoD velocity curve of a centre channel while 16 neighbours are jetting at several time delays. As depicted, the drop velocity is drastically decreased due to the crosstalk between the channels. A time delay between the odd and even channels is considered and the time delay is varied from 0 to 5 μs. The effect of crosstalk can be effectively reduced by the time delay, see Fig. 3.25. As shown, the crosstalk effect depends on the time delay. When a proper value for the time delay is chosen, for instance 3 μs, the deviations in the drop velocity are limited. Note that a time delay of 0 μs shows the influence of the crosstalk on the drop velocity. The drop velocity is considerably reduced due to the crosstalk.

An optimisation problem is formulated to obtain the optimal time delay between the odd and even channels to minimise the effect of the crosstalk. Since no precise model is available to describe the dynamics of the crosstalk between the channels, the optimisation problem is formulated based on the measured data that are provided by the high-speed camera. The objective is to minimise differences between drop velocity of the odd and even channels at all jetting frequencies.

Define the optimisation problem as

$$\mathscr{F} = \sum_{f=F_{\min}}^{F_{\max}} \sum_{t=0}^{T} (v_{\text{odd}}(t,f) - v_{\text{even}}(t,f))^2, \tag{3.17}$$

subject to

$$T_{D_{\min}} \leq T_D \leq T_{D_{\max}}, \tag{3.18}$$

where f is the jetting frequency, v_{odd} is the odd channels' average drop velocity, v_{even} is average drop velocity of the even channels, T_D denotes the time delay between the jetting of the odd and even channels, T denotes the total time of the experiment, and t is the time instance when the measurements are taken. The optimal time delay that minimises the cost function is given by

$$T_{D_{\text{opt}}} = \arg\min_{T_D} \mathscr{F}. \tag{3.19}$$

This optimisation problem is a constrained non-linear optimisation, which is solved using a standard optimisation solver. The optimisation problem is solved with the printhead in the loop. The results of this optimisation problem are evaluated in the next subsection.

3.6.2 Experimental Results

The optimised pulse is used to jet several bitmaps, a sample of these bitmaps is shown in Fig. 3.26 moving top down in time with a jetting frequency of 48 kHz. The time history of the drops travelling from the nozzle plate to the print media are collected to analyse the performance of the printhead. The drop velocity of the jetted bitmap using the optimised and the standard pulse are respectively shown in Figs. 3.27 and 3.28.

The performance is evaluated based on the maximum drop velocity variation of the jetted drops. The maximum drop velocity variation is less than 1 m/s for the optimised pulse, while the variation is 2 m/s for the standard pulse. The overall

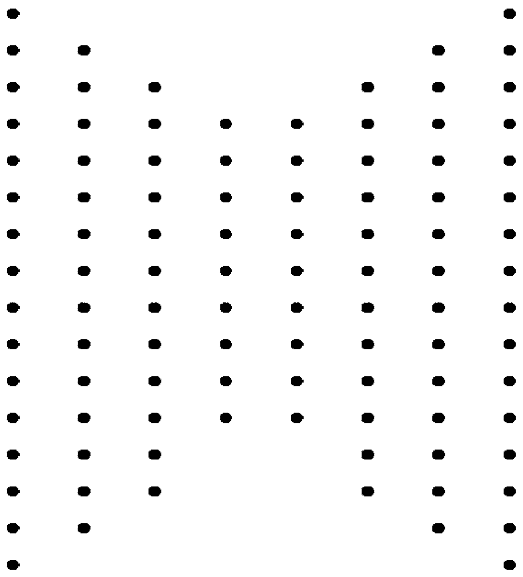


Fig. 3.26 The bitmap to be printed. In this bitmap, we consider 8 nozzles jetting different number of drops at a DoD frequency of 48 kHz

improvement in the drop velocity consistency achieved using the optimised pulse has a considerable influence on the print quality as depicted in Figs. 3.29 and 3.30. Figure 3.29 shows the printed bitmap, which has a regular pattern compared to Fig. 3.30. The improvement in the drop velocity consistency, which is achieved using the optimised pulse, has a positive influence on the print quality. On the other hand, the large variations in the drop velocity, in case of the standard pulse, result in an unacceptable poor print quality. Figure 3.30 shows that the first drop is faster than the subsequent drops and several drops are merged and form drops with large volume. The original pattern cannot be retrieved from the printed bitmap.

3.6.3 Bitmap-Based Adaptation

For any asymptotically stable non-linear system the output, y, is periodic if its input, u, is periodic. The trajectory in the phase plane of the meniscus position and meniscus velocity is input frequency dependent. For continuous jetting (so all 1's), the phase plane of the meniscus velocity and meniscus position will converge to a stable limit cycle. That limit cycle generates stable drops with stable volume and velocity, see Fig. 3.31. When the jetting pattern includes a non-jetting sample, a 0-pixel, the periodic behaviour is destroyed, see Fig. 3.32, and thus the stable limit cycle. It requires several jetting drops to converge again to the stable limit cycle. The convergence rate depends on the dynamics of the channel and of the input frequency. The measurements show that after 100 µs the transient is damped out.

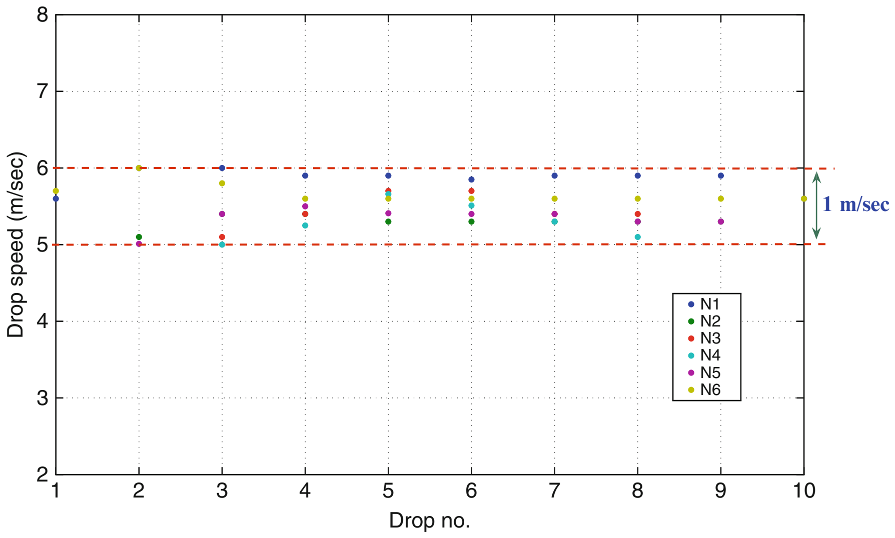


Fig. 3.27 Drop velocity of the jetted bitmap for the 8 nozzles using the optimised pulse. The drop velocity variation is kept less than 1 m/s

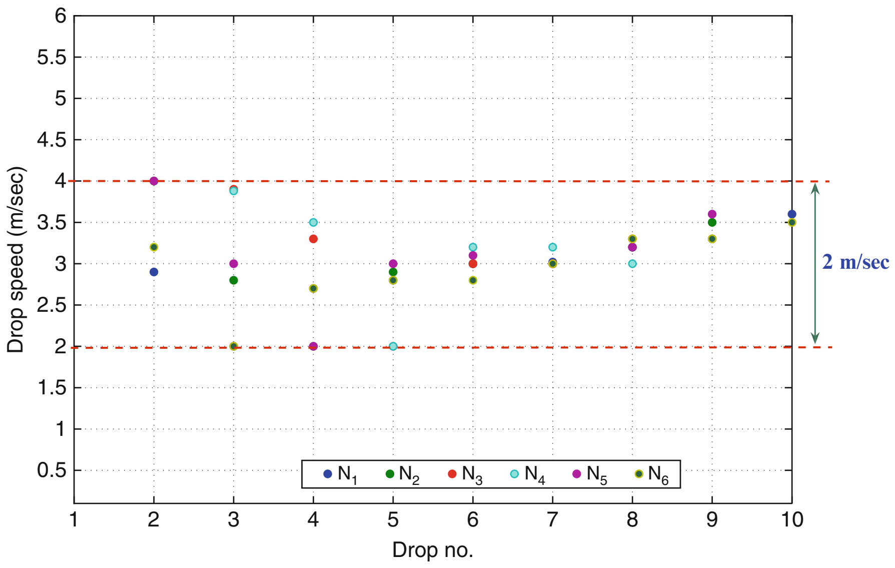


Fig. 3.28 Drop velocity of the jetted bitmap for the 8 nozzles using the standard pulse. The drop velocity variation is around 2 m/s

Thus, with a jetting frequency above 10 kHz, there will always be transients. As jetting frequencies up to 100 kHz are desired for higher throughput/accuracy, any design of a pulse shape has to deal explicitly with these transients.

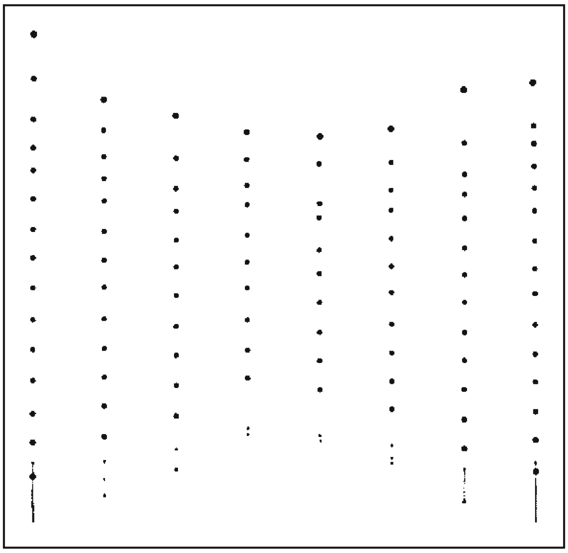


Fig. 3.29 Printed bitmap using the optimised pulse moving top down in time

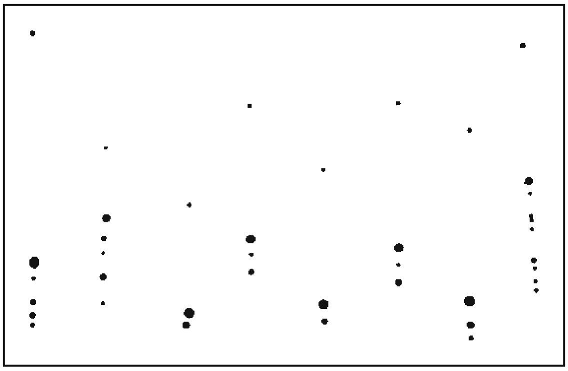


Fig. 3.30 Printed bitmap using the standard pulse. The variation in the drop velocity results in a poor print quality. The first drop is faster than the subsequent drops, several drops are merged together and form drops with large volume

Our proposal is to sustain the limit cycle to preserve the correct values of the initial conditions (meniscus position and velocity) when a drop will be fired, independent whether the previous pixel is on or off. Toward this objective, we propose a non-jetting pulse for the 0-pixel just before jetting the 1-pixel. This non-jetting pulse ensures that the trajectory in the phase plane is at same position as when a drop is being jetted. The meniscus motion is drawn in the phase plane in Figs. 3.31 and 3.32. Along the axes position and velocity are shown which allows good analysis of limit cycle behaviour.

The trajectories shown are based on a wave model. In this model, the behaviour of the printhead is described with travelling waves, which is used to solve the differential equations implicitly. The drop formation is calculated simultaneously with the wave transport. This is a good link between the wave reflection at the nozzle

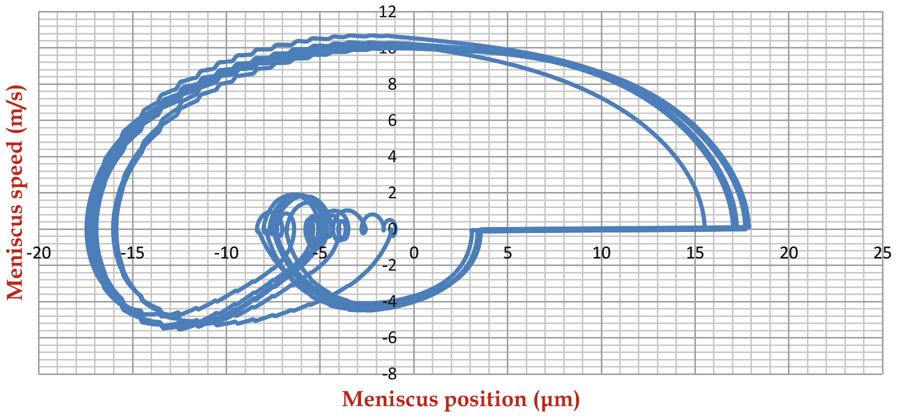


Fig. 3.31 Phase plane for continuous jetting

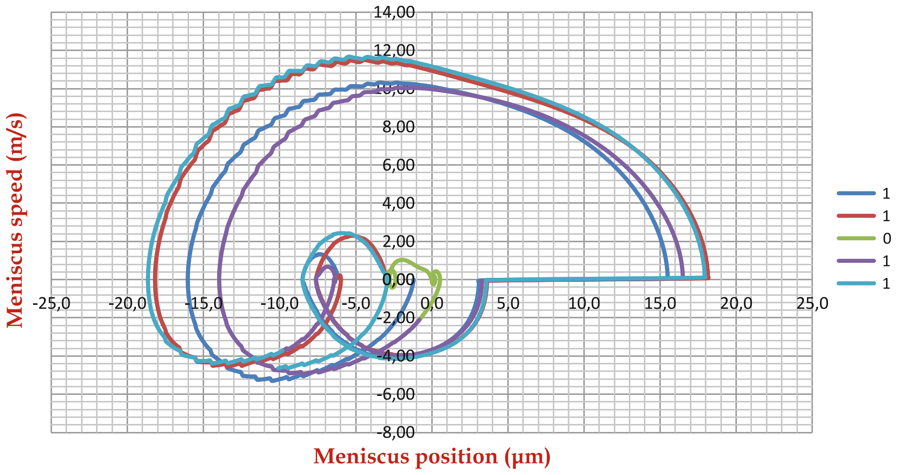


Fig. 3.32 Phase plane for jetting pattern 11011

and the flow in the nozzle. The main advantage of this approach is that it takes into account the time-varying degree of filling of the nozzle. Moreover, this model can predict the meniscus state (velocity and position), which is useful to understand the behaviour of the printhead. We have used this model to have more insight into the physics of the printhead.

As explained, the drop properties depend on the bitmap which is printed. Therefore, based on the bitmap the input pulse is updated. For example, a 0-pixel in the pattern will result in succeeding drops with different properties. The input pulse has to be updated based on a number of preceding pixels in the pattern.

In our optimisation problem, we consider a 4-pixel pattern and we optimise the input pulses for the fourth drop taking into account all eight possible patterns of

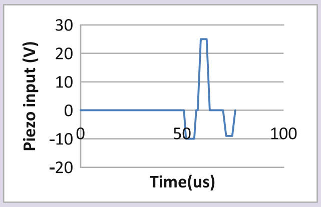
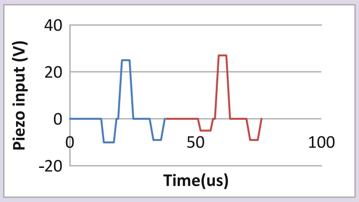
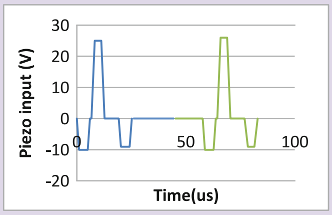
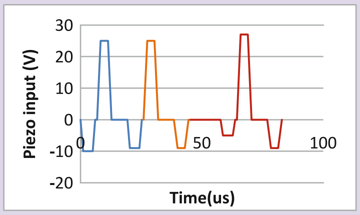
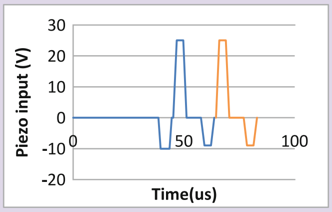
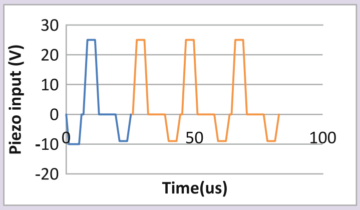
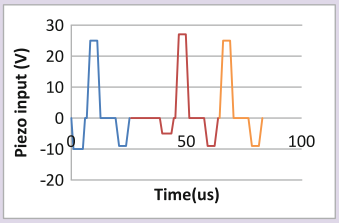
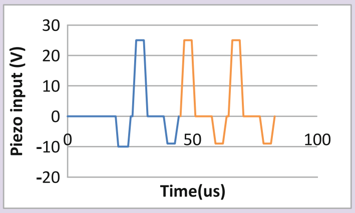
Pattern	0001 (P3)	0101 (P1)
Input pulses		
Pattern	1001 (P2)	1101 (P1)
Input pulses		
Pattern	0011 (P0)	1111 (P0)
Input pulses		
Pattern	1011(P0)	0111 (P0)
Input pulses		

Fig. 3.33 Optimised pulses for different jetting patterns. Each of the four different jet pulses, P*i*, is given its own colour

the preceding three pixels. As we assume the drop velocity of the fourth pixel only depends on the number of preceding 0-pixels, the optimisation is simplified to only four cases. The four cases are 0001, 1001, x101, and xx11. We start with the pattern 0001 and we optimise the input of the fourth drop such that the jetted drop has a

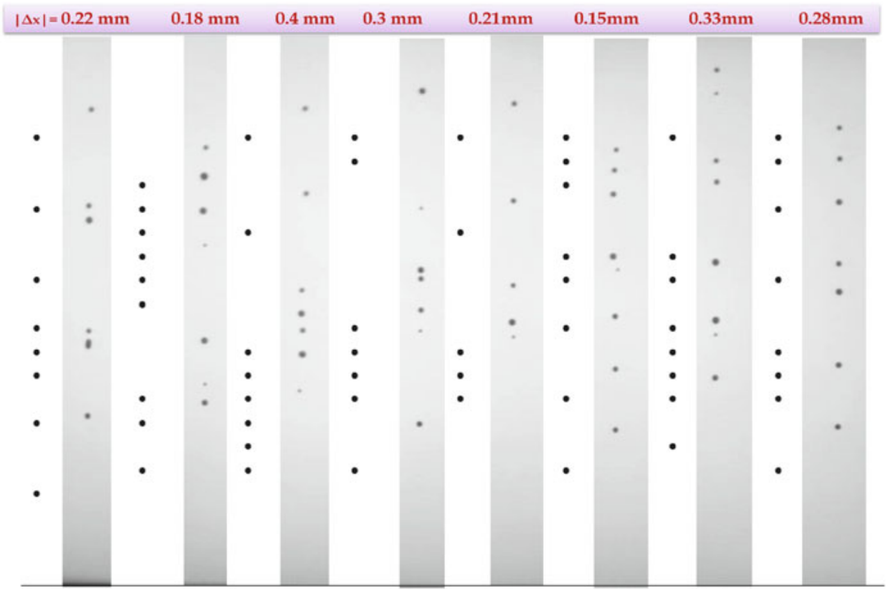


Fig. 3.34 Printed bitmap using the standard pulse (desired bitmap, printed bitmap (*grey*))

desired velocity. The resulting optimised pulse of the pattern 0001 is considered as an input for the first drop for the subsequent optimisation problems. Figure 3.33 summarises the optimised pulses and the corresponding patterns. We can distinguish four different pulses depending on the number of preceding 0-pixels.

In Figs. 3.34 and 3.35, we compare the print quality when jetting an arbitrary bitmap with a basic jetting frequency of 52 kHz using the standard pulse and the optimised pulses, respectively.

The standard pulse shows a poor print quality and many drops are merged together and the jetting pattern cannot be retrieved. Using the optimised pulse with pattern adaptation, the quality of the print is improved. Thanks to the non-jetting pulse, there are no large differences in drop velocity between the low- and high-frequency drops. The maximum position error is improved to 0.15 mm compared to 0.4 mm for the standard pulse assuming a predefined paper distance of 2 mm.

To summarise, in this section, we have demonstrated that dividing the channels into a number of groups and introducing a proper time delay between the actuation of the channels are sufficient to cope with the effect of the crosstalk between the channels. Further, we have shown that the DoD velocity curve optimisation is not enough to improve the print quality since it neglects essential effect of the 0-pixel in bitmap. The state of the meniscus (velocity, position) at the start of the pulse influences the drop velocity considerably. Any pulse design has to guarantee almost the same initial meniscus state at firing a drop. For jetting patterns containing 0, a non-jetting pulse is introduced to improve the performance of the subsequent drops.

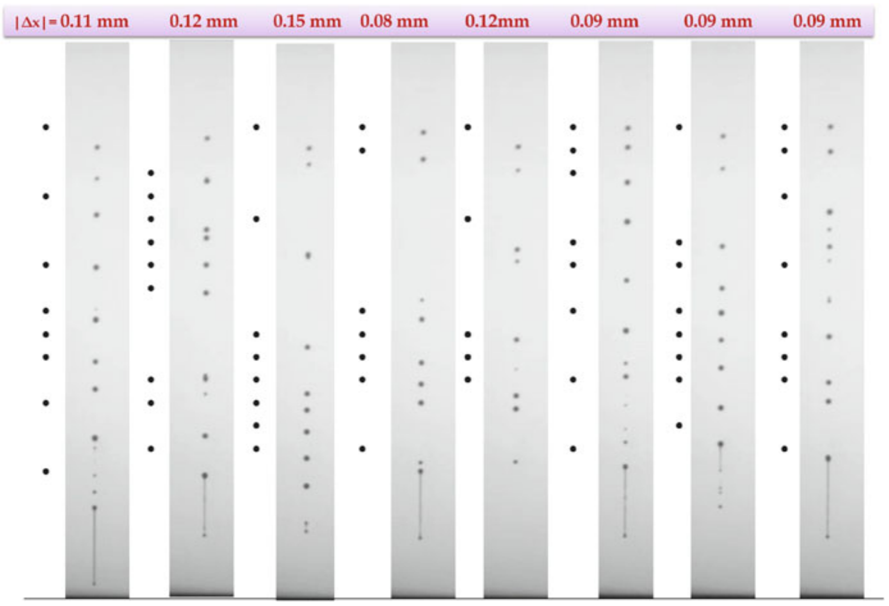


Fig. 3.35 Printed bitmap using the optimised pulses (desired bitmap, printed bitmap (*grey*))

Based on the jetting bitmap, a set of input pulses is optimised. The use of these pulses results in smaller drop velocity variations for random bitmaps and, therefore, better print quality.

3.7 Discussion and Conclusion

The inkjet printhead is the core component in an important number of print platforms now and the relevance of this print technology is expected to grow further in the future. Achieving more complete control over the imaging device is required to attain variable-speed colour printing with acceptable print quality and to enable the many emerging industrial applications. This involves drop size, velocity, and timing and the challenge was to reach this goal using smart feedforward control with a printhead as is.

In this section, answers for the following questions are discussed.

- To what extent has the problem been solved?
- Where and when can results be applied?
- How do they complement each other?
- What will be the future directions?

Compared to a standard input jetting pulse, large improvements have been achieved towards enabling fully variable-speed printing. Drop velocity variations grow with more than an order of a magnitude in this setting. With the techniques presented, these variations can almost be limited to the allowable tolerances of the conventional fixed printing speed mode. Moreover, using these techniques in this conventional setting, the robustness of the system will also be enlarged considerably.

Two approaches have been presented. The first approach of experiment-based control involved the use of a camera to inspect drop characteristics. This is applicable in the design phase and even in manufacturing. Techniques to deduce drop characteristics from the camera images can also be used for inspection. By parameterising the input pulse using the basic characteristics of the frequency response, a low-complexity pulse shape results. This pulse shape can be used in various, also in-line, optimisation loops. The research also showed that it is usually sufficient to introduce a group delay to minimise the crosstalk. Incorporating only one additional parameter, a delay, in the optimisation allows much simpler extension to the MIMO case. Extension of the input parametrisation with the possibility of using non-jetting pulses to improve transient behaviour allows us to optimise over the actual print, the bitmap to be realised.

The second approach shows that without a camera, by using model-based control, drop characteristics can still be improved strongly. The model-based approach provides measures for on-line adaptation. Especially when the uncertainty is taking into account explicitly, robust performance can be optimised. Additionally, also here, intermediate results in applying this technique provide valuable information. Thorough methods to identify the process dynamics, the process dependency on its point of operation, and the corresponding uncertainty are important in system analysis, also for newly built or adapted printheads.

This research also leaves a number of open issues and interesting research directions for the future to further enhance control over the drop properties of an inkjet printhead. Next to further elaborating on the lines presented here, especially lifting some of the constraints put on this research line will enable additional improvement. The constraints mainly came from the limitations of current printhead components. As the ability of these components is rapidly growing, it can be expected that in the near future many of these possible enhancements will become reality. Without being complete, some interesting directions can be found in

- *Enhanced models for control.* The limitations of current physical models for proper control have been discussed. Improved models with limited complexity describing all the relevant phenomena for control and derived from either physical laws, experiments, or even better an optimal combination of both, will allow more performance and robustness achievable with control.
- *Drivers allowing more freedom in the design of the actuation pulses.* The input pulse design has been performed for pulses with low complexity. As shown, this can be very effective to deal with the most prominent physical properties. Further optimisation will require more complex input pulses and cost-effective and energy-efficient drivers will be necessary to use those inputs in practice.

- *Enabling the use of additional, possibly real-time and in-line, sensors*. Additional sensors will allow more effective measurement of the relevant parameters to be controlled in the system. By restricting input design to feedforward only, a very limited number of control techniques can be used. Real-time in-line sensors open up the possibility for feedback control.
- *Fast feedback components*. Until now it has not been possible, or more precisely, not cost effectively possible to use feedback for inkjet printheads due to the relevant time scales in these systems. The development of every time faster and more reasonably priced components in combination with proper sensing, will open up possibilities for the use of feedback. Feedback can more effectively deal with variations in the process characteristics and the disturbances acting upon the system.
- *Joint design of printhead and its control*. Often many jet aspects in the printhead have already been optimised in the design of the printhead itself without considering more freedom in the design of the input pulse. Sub-optimal behaviour results and this can be prevented for by early incorporation of control freedom in the design.

Many interesting future directions are still open to elaborate on but all together one can conclude that large steps have been made towards the ability to jet any drop at any time with inkjet printheads using a systems and control approach. This has brought us closer to the point where we can use these components fully effectively in variable-speed colour printing systems. Future enhanced inkjet imaging devices, which most efficiently make use of the potential of control, will appear to attain the highest performance, robustness, and cost effectiveness. The design techniques presented here allow us to balance these different aspects.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. Bartholomew-Biggs, M.: Nonlinear Optimization with Engineering Applications. Springer Optimization and its Applications, vol. 19. Springer, New York (2008)
2. Beltman, W.M.: Viscothermal wave propagation including acousto-elastic interaction, part I: theory. J. Sound Vib. **227**, 555–586 (1999)
3. Berger, S.S., Recktenwald, G.: Development of an improved model for piezo-electric driven ink jets. In: Proceedings of the 19th IS&T International Conference on Digital Printing Technologies (NIP19), New Orleans, pp. 323–327 (2003)
4. Bertsimas, D., Brown, D.B., Caramanis, C.: Theory and applications of robust optimization. SIAM Rev. **53**, 464–501 (2011)
5. Dijksman, J.F.: Hydrodynamics of small tubular pumps. J. Fluid Mech. **139**, 173–191 (1984)
6. Ezzeldin, M., van den Bosch, P.P.J., Jokic, A., Waarsing, R.: Model-free optimization based feedforward control for an inkjet printhead. In: Proceedings of the 2010 IEEE International Conference on Control Applications (CCA 2010), Yokohama, pp. 967–972 (2010)

7. Ezzeldin, M., van den Bosch, P.P.J., Weiland, S.: Improving the performance of an inkjet printhead using model predictive control. In: Proceedings of the 18th IFAC World Congress (IFAC 2011), Milano, pp. 11544–11549 (2011)
8. Ezzeldin, M., van den Bosch, P.P.J., Weiland, S.: Towards a better printing quality for an inkjet printhead. IEEE Control Syst. **33**(1), pp. 42–60 (2013)
9. Georgakis, C.: A model-free methodology for the optimization of batch processes: design of dynamic experiments. In: Proceedings of the 7th IFAC International Symposium on Advanced Control of Chemical Processes (ADCHEM 2009), Hong Kong, pp. 644–649 (2009)
10. Golub, G.H., van Loan, C.F.: Matrix Computations. John Hopkins University Press, Baltimore (1989)
11. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley, Boston (1993)
12. Groot Wassink, M.B.: Inkjet printhead performance enhancement by feedforward input design based on two-port modeling. Ph.D. thesis, Delft University of Technology, Delft (2007)
13. Hutchinson, S., Hager, G.D., Corke, P.I.: A tutorial on visual servo control. IEEE Trans. Robot. Autom. **12**, 651–670 (1996)
14. Khalate, A.A., Bayon, B., Bombois, X., Scorletti, G., Babuška, R.: Drop-on-demand inkjet printhead performance improvement using robust feedforward control. In: Proceedings of the 50th IEEE Conference on Decision and Control (CDC 2011), Orlando, pp. 4183–4188 (2011)
15. Khalate, A.A., Bombois, X., Babuška, R., Scorletti, G., Koekebakker, S., Wijshoff, H., de Zeeuw, W., Waarsing, R.: Performance improvement of a drop-on-demand inkjet printhead: a feedforward control based approach. In: Proceedings of the 27th IS&T International Conference on Digital Printing Technologies (NIP27), Minneapolis, pp. 74–78 (2011)
16. Khalate, A.A., Bombois, X., Babuška, R., Wijshoff, H., Waarsing, R.: Performance improvement of a drop-on-demand inkjet printhead using an optimization-based feedforward control method. Control Eng. Pract. **19**, 771–781 (2011)
17. Khalate, A.A., Bombois, X., Scorletti, G., Babuška, R., Waarsing, R., de Zeeuw, W.: Robust feedforward control for a drop-on-demand inkjet printhead. In: Proceedings of the 18th IFAC World Congress (IFAC 2011), Milano, pp. 5795–5800 (2011)
18. Khalate, A.A., Bombois, X., Ye, S., Babuška, R., Koekebakker, S.: Minimization of cross-talk in a piezo inkjet printhead based on system identification and feedforward control. J. Micromech. Microeng. **22**, Paper 115035 (2012)
19. Ljung, L.: System Identification – Theory for the User. Prentice Hall, Upper Saddle River (1999)
20. MATLAB: http://www.mathworks.com/products/matlab/. Accessed Aug 2012
21. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
22. Ruszczynski, A.P.: Nonlinear Optimization. Princeton University Press, Princeton (2006)
23. Scherer, C., Weiland, S.: Linear Matrix Inequalities in Control. Lecture Notes. DISC, Delft (2006)
24. Schield, T.W., Bogy, D.B., Talke, F.E.: A numerical comparison of one-dimensional fluid jet models applied to drop-on-demand printing. J. Comput. Phys. **67**, 327–347 (1986)
25. Slot, M., Feenstra, F., de Witte, P.: Roadmap printing: from the world of print to the printed world. In: High Tech Systems and Materials Platform, Eindhoven (2011)
26. The Mathworks Inc.: Matlab Optimization Toolbox. The Mathworks Inc., Natick (2010)
27. Tijdeman, H.: On the propagation of sound waves in cylindrical tubes. J. Sound Vib. **39**, 1–33 (1975)
28. van den Hof, P.M.J., Bombois, X.J.A.: System Identification for Control. Lecture Notes, DISC, Delft (2007)
29. van der Heijden, F.: Image Based Measurement Systems: Object Recognition and Parameter Estimation. Wiley, Hoboken (1995)
30. Wijshoff, H.M.A.: The dynamics of the piezo printhead operation. Phys. Rep. **491**, 77–177 (2010)
31. Zhou, K., Doyle, J.C.: Essentials of Robust Control. Prentice Hall, Eaglewood Cliffs (1998)

Chapter 4
Adaptive Control Strategies for Productive Toner Printers

Paul van den Bosch, Carmen Cochior, Mohamed Ezzeldin, Perry Groot, Peter Lucas, Jacques Verriet, René Waarsing, and Siep Weiland

Abstract This chapter discusses design considerations for industrial systems and processes when embedded systems allow to intelligently influence the system in real-time. It is shown that in such embedded systems the capability to adapt themselves to changing environments and/or to different operating conditions has to be exploited. If properly done, almost all performance indicators like accuracy, speed, robustness, insensitivity for disturbances, will improve. The challenge is to first study the process for which the behaviour has to be improved. Based on the characteristics of the process and its disturbances, one wishes to select among the hundreds of tools to achieve its goals. For the professional printer, this design process and its many compromises and choices will be illustrated. Consequently, the professional printer will be analysed for its specific characteristics whose behaviour can be influenced during printing. Based on these characteristics, appropriate control approaches are discussed in more detail to show how control can cope with uncertainty or changing parameters. With extensive and illustrative examples the

P. van den Bosch (✉) • C. Cochior • M. Ezzeldin • S. Weiland
Control Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: p.p.j.v.d.bosch@tue.nl; c.cochior@tue.nl; m.ezz@tue.nl; s.weiland@tue.nl

P. Groot • P. Lucas
Department of Model-Based System Development, Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
e-mail: perry@cs.ru.nl; peterl@cs.ru.nl

J. Verriet
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: jacques.verriet@esi.nl

R. Waarsing
Océ-Technologies B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands
e-mail: rene.waarsing@oce.com

various methods are compared and it is shown why some control approaches are preferred solutions in the large number of problems that are faced by professional printers.

4.1 Introduction

Dealing with uncertainty in physical processes can be solved in several ways. One way is to assume that the process behaves reasonably well and behaves more or less predictable. Then, a deterministic, nominal model with all uncertainty mapped in unknown disturbances can be derived either by using first-principle physical knowledge or by estimating an appropriate model based on measurements from appropriately selected experiments performed on that system. Another approach is needed when it is assumed that a deterministic model is unfeasible. In that case, all we have are measurements of a complex, unpredictable process. Based on these measurements, an attempt can be made to describe the process behaviour with a stochastic and/or statistical model, such as a Gaussian stochastic model or as a statistical Markov chain.

In both cases, control is an excellent approach as it delivers high performance to systems even when parameters of the system change and/or disturbances start to deteriorate system behaviour.

Basically, accurate models and/or accurate measurements are needed to guarantee that the output of a process satisfies the required values. Let us first introduce some variables to show the consequences of modelling and measurement errors. We define an input u, output y, disturbance w, and a reference value for the required output r. Let us assume that the process can be accurately represented as H_0, that we can find an approximate model H of the process and that we have designed a controller C. The measured output y of the real process will be $y = H_0 \cdot u + w$. That is, the measurement is a noise-corrupted output of the process that is controlled by the input u. If we rely on the knowledge of the approximate model H, an appropriate input u for the process that achieves the reference r as its output could be $u_{\mathrm{FF}} = H^{-1} \cdot r$, where the subscript FF indicates feedforward. Now the actual output of the process will become:

$$y = H_0 \cdot u_{\mathrm{FF}} + w = H_0 \cdot (H^{-1} \cdot r) + w. \tag{4.1}$$

It can be observed that when we accurately know the process, so $H = H_0$, and there are no disturbances, so $w = 0$, this feedforward value u_{FF} of u, yields that the output equals the reference value, so $y = r$. To establish this, no measurements are needed, ONLY an accurate model H of H_0 and NO disturbance. However, this chapter deals with systems and processes that are not accurately known or are time varying during the operation of the system, so $H_0 \neq H$, and when disturbances act on the process, so $w \neq 0$. Consequently, another approach has to be taken as illustrated in Fig. 4.1.

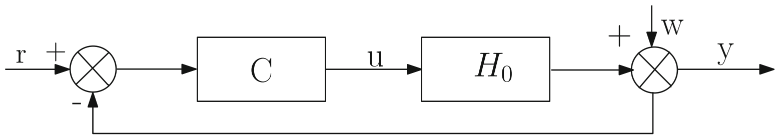


Fig. 4.1 Closed-loop control system

We can no longer rely only on accurate models. Measurements have to be used too. When using measurements of the output we can calculate the error e between the required value r and the actual value y, with $e = r - y$. If we use a feedback controller C to calculate an appropriate value of the input u for the process based on the error e, the control input u will be $u = C \cdot e$, and we will have the following relations:

$$y = H_0 \cdot u + w = H_0 \cdot C \cdot e + w, \tag{4.2}$$

$$y \cdot (1 + H_0 \cdot C) = H_0 \cdot C \cdot r + w, \tag{4.3}$$

$$y = \frac{H_0 \cdot C}{1 + H_0 \cdot C} \cdot r + \frac{1}{1 + H_0 \cdot C} \cdot w. \tag{4.4}$$

When properly designed, the value of the controller C will be (very) large within the area we want to control the process, yielding $H_0 \cdot C \gg 1$. Consequently $y = r$. This last result shows the power of control when both the model of the process is not correct ($H_0 \neq H$) and disturbances act on the process ($w \neq 0$). When accurately measured and an appropriate controller is designed, we can guarantee that the output y of the process will follow the required value r. In this setting, control looks like magic. It is not. Two major requirements are posed:

- A sufficiently accurate, but not exact, model is needed to design the controller C. Errors of 10–50% are allowed, but the basic dynamics have to be captured correctly.
- The measurements have to be accurate. If measurement noise is present, the control performance deteriorates accordingly.

Within the control area several approaches exist to cope with different classes of uncertainty and disturbances with advanced feedback controllers. These approaches are called Optimal, Robust, and Adaptive control. We discuss these approaches first.

- In an *optimal control* approach, the best controller is designed based on accurate knowledge of the process and its environment. If the uncertainties are explicitly known, the best controller is designed. The performance will be high, but if the assumptions about process or environment are not correct, the performance will deteriorate quite fast. It is not the best way to deal with uncertainty or changing operating conditions when uncertainty is dominant.

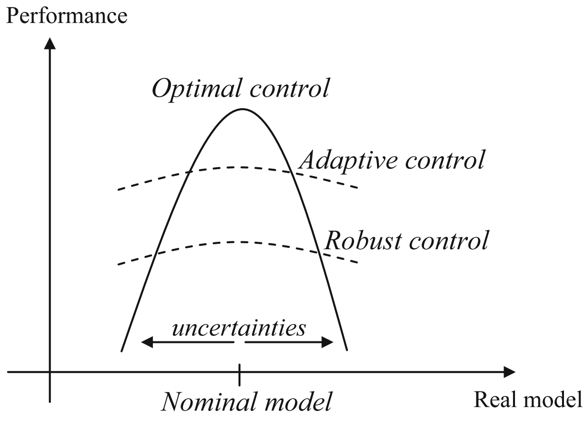


Fig. 4.2 Qualitative comparison of Optimal, Robust, and Adaptive control

- In a *robust design* the goal is to predict all possible deviations that can occur. Then, a controller is synthesised that can still deliver an acceptable performance even in the worst situation. So, the design is inherently conservative. There is always a trade-off between robust stability and robust performance. Therefore the price to be paid for increased robustness with respect to stability is a reduction in performance.
- An attractive alternative of robust control is *adaptive control*. Then, the controller explicitly or implicitly estimates the changing parameters or changing disturbances and adapts itself to the newly estimated situation. Within the new situation it can still achieve high performance. As it adapts itself to changing parameters and disturbances, it can be shown to be robust. In that sense adaptive control combines the advantages of optimal control (high performance) with those of robust control (insensitive for a changing environment). However, adaptive control requires an intensive on-line computation and it is sensitive to measurement noise.

In Fig. 4.2 the relative behaviour of Optimal control, Robust control, and Adaptive control is elucidated. It represents the expected performance of those controllers as functions of the changes in parameters and disturbances.

In the sequel, we will discuss how to select appropriate design approaches for a professional printer.

4.2 Characteristics of an Industrial Electro-Photography Printer

In this section, we try to describe and analyse the characteristics of industrial toner-based printers. Knowing these characteristics it is possible to select appropriate control approaches. First a short description of the physics behind electro-photography printers.

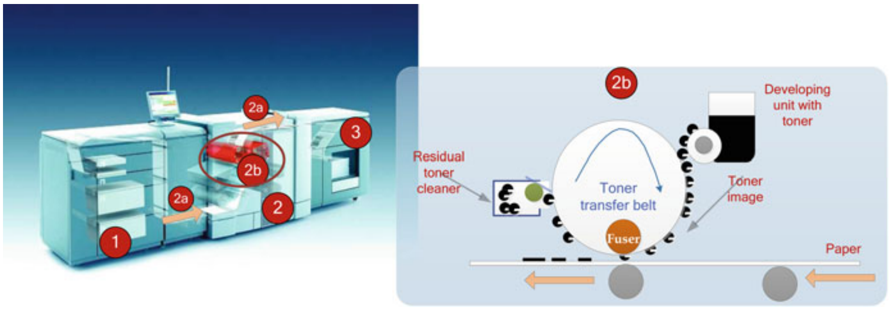


Fig. 4.3 Electro-photography printer: *1*. Paper input; *2*. Print engine: *a*. Paper path, *b*. Printing process; *3*. Finisher

4.2.1 Heat Flow Model

The printing process consists of six main steps: (1) charging of a photo-conductor, (2) exposing the photo-conductor drum or belt to the image, (3) development of the latent image, (4) transferring the image from the photo-conductor to a sheet of paper or any other printing media, (5) fusing the developed image to the printing media, and finally (6) cleaning any residual toner from the photo-conductor drum or belt in preparation for the next print [6, 13, 30]. These main steps are illustrated in Fig. 4.3. From the input tray the paper is transported first via a preheater along the toner belt with toner to the fusing point. Subsequently, it is transported to the output tray.

An important step, and the topic of this chapter, is the thermal process to fuse the toner on the paper. The temperature of fusing is very important. When this temperature is too low, the toner will not penetrate the paper and can easily be removed from the paper. If the temperature is too high, the toner will melt to any surface and, consequently, not always at the intended positions on the paper.

The thermal behaviour of a toner-based printer can be modelled with two elements, the thermal capacity C [J/K] and the thermal resistance R [K/W]. The thermal capacity is a buffer of the thermal energy $E = C \cdot T$ [J] with the temperature T [K] as variable which determines the energy stored in that buffer [20]. Basically, the following equations are valid with P [W] the thermal power flow. The time derivative of the temperature is proportional to the net incoming power flow P and the temperature difference is also proportional to the power flow P through the thermal resistor R:

$$\dot{T} = \frac{1}{C}P,$$
$$T = RP. \tag{4.5}$$

Here, $\dot{T}$ stands for $\frac{dT}{dt}$, the time derivative of the temperature.

In the printing system, thermal power (heat) is transported in three different ways, namely, conduction, convection, and radiation. The heat is transported by conduction through the mechanical components. Heat flow is convected via air flow, and radiated by hot surfaces. In addition, heat is transported via the movement of objects in the printer, such as the printing media or the transport system.

Heat conduction and convection are modelled as linear thermal resistances, where the resistance may depend on system parameters such as rotational speed [16]. Heat radiation is modelled as a non-linear resistance. Heat, which is inserted into the printer, is represented as electric power. The printing system, which we study in this chapter, has two heaters that generate heat from electrical energy. The preheater, P_{pre}, is responsible for heating up the paper to a desired temperature before fusing. The toner transfer belt (TTF) rollers are usually heated using a tungsten quartz lamp P_{TTF}.

The following assumptions are made to derive the thermal model (4.6) below.

- The printer is modelled as a lumped component, where the description of the behaviour of the spatially distributed thermal system is simplified by an electrical circuit consisting of discrete entities that approximate the behaviour of the distributed system under certain assumptions. This approximation is useful to simplify complex differential heat equations. In this model, only the most interesting temperatures are explicitly included.
- The sheets are not modelled individually but as a continuous paper mass flow, which interacts with the preheater and the TTF belt.
- The interaction between areas of different temperatures are modelled using thermal resistance R between the temperatures of the lumped thermal capacities C.
- The printing media (the paper) extracts heat from the preheater and exchanges heat with the TTF.

A schematic view of the process can be seen in Fig. 4.4, where the printing media size is given by a length l [m] and a width b [m], while d [m] denotes the distance between two printing media.

As mentioned, controlling the fuse temperature is the key to achieve high print quality. The fuse temperature T_{fuse} is determined by the temperature of the paper sheets and the temperature of the TTF belt at the fuse pinch. The fuse temperature cannot be measured. Therefore, the preheating temperature T_{pre} and the TTF temperature T_{TTF} are used as a good estimation of the fuse temperature. The dynamics of the preheating and TTF systems are given by the following lumped model [16, 17].

$$
\begin{aligned}
\dot{T}_{\text{pre}} &= \frac{1}{C_{\text{pre}}}\left(P_{\text{pre}} - m_{\text{pap}} v c_{\text{pap}} (T_{\text{pap}} - T_{\text{init}}) - \frac{T_{\text{pre}} - T_{\text{env}}}{R_{\text{env}}}\right), \\
\dot{T}_{\text{TTF}} &= \frac{1}{C_{\text{TTF}}}(\eta_{\text{TTF}} P_{\text{TTF}} - \frac{T_{\text{TTF}} - T_{\text{env}}}{R_{\text{env}}} - \frac{T_{\text{TTF}} - T_{\text{pap}}}{R_{\text{pap}}} \\
&\qquad - \frac{T_{\text{TTF}} - T_{\text{slow}}}{R_{\text{slow}}} - \frac{T_{\text{TTF}} - T_{\text{roller}}}{R_{\text{roller}}}),
\end{aligned}
$$

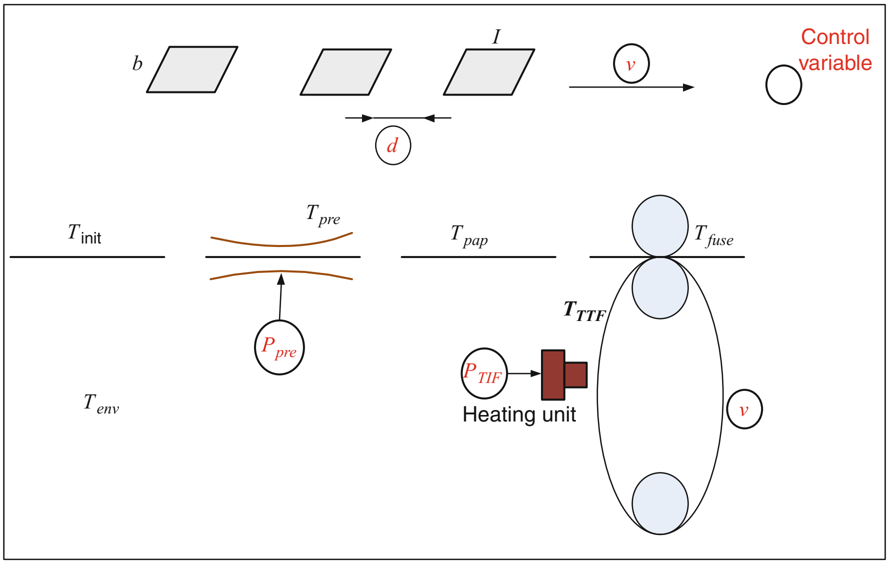


Fig. 4.4 Schematic view of the printer hardware

$$\dot{T}_{\text{roller}} = \frac{1}{C_{\text{roller}}}\left(\frac{T_{\text{TTF}} - T_{\text{roller}}}{R_{\text{roller}}}\right),$$
$$\dot{T}_{\text{slow}} = \frac{1}{C_{\text{slow}}}\left(\frac{T_{\text{TTF}} - T_{\text{slow}}}{R_{\text{slow}}}\right), \tag{4.6}$$

with

$$T_{\text{pap}} = T_{\text{init}} + (T_{\text{pre}} - T_{\text{init}})(1 - e^{\frac{-\lambda L_{\text{pre}}}{m_{\text{pap}} v}}),$$

where T_{init} is the paper temperature when entering the printer, C [J/K] represents the thermal capacity of the corresponding object, R [K/W] is the thermal resistance of the corresponding object, P_{pre} [W] and P_{TTF} [W] denote the input power to the preheater and the TTF heater, respectively, η_{TTF} [$-$] denotes the TTF heater thermal efficiency, m_{pap} [kg/m] is the paper mass per unit length, v [m/s] is the belt speed, T_{pap} [K] is the average temperature of the paper, and L_{pre} [m] represents the length of the preheater. T_{roller} [K] and T_{slow} [K] represent the temperatures of the various rollers that are in contact with the TTF. T_{env} [K] denotes the temperature of the environment to which thermal power is leaked.

An estimation formula is used to estimate the fuse temperature. The fusing system can be described by

$$T_{\text{fuse}} = h(T_{\text{pre}}, T_{\text{TTF}}, P_{\text{pre}}, P_{\text{TTF}}, v, d). \tag{4.7}$$

Due to reasons of industrial confidentiality, the details of the fusing model are omitted here. The model (4.6) is connected to a higher-level controller, which determines the amount of available power, the printer speed, and the distance between sheets. Several parameters of the model (4.6) are time varying, since these parameters depend on the different printing jobs and the rotation speed, while other parameters are unknown.

4.2.2 Characteristics Determining the Control Problem

In the previous section, a deterministic lumped model of the electro-photography printer has been formulated. It is only valid when all the stated assumptions are true and all parameters are known. A priori, we know that this model is a valid but rough approximation, as the assumptions may not be valid in practice. But, as described in the previous section, when applying control it is possible to cope with these model deviations and unmodelled disturbances. First we will describe the disturbances w acting on the printer and that influence the fuse temperature.

The following disturbance components can be distinguished in the variable w:

- The print jobs defined by the user require different paper sheets, each represented by its thermal and mechanical characteristics such as mass [gr/m^2], size, humidity, surface, and initial temperature. Between jobs, but also within a print job, different sheets of paper can be selected. That indicates that the temperatures in the printer are highly influenced by the arrival of sheets into the printer. Heavy and large sheets require much more thermal power (and so electric power for the heaters) to reach the required temperature of the fuse. From a control point of view these print jobs, stored in the *print queue*, will disturb the thermal process considerably. The changes are fast (changing stepwise) and large. Owing to the large thermal capacities in the printer and the limited amount of electrical power, it is difficult to cope with these large changes. However, there is one big advantage. The print jobs in the print queue are known in advance. It is possible to anticipate the disturbing effect of the different sheets in the print queue. When more heavy sheets are expected, the temperatures can already be raised to cope with their higher demands for thermal power in the future. The major influence of different sheets of paper are changing parameters of the printer (thermal capacities and resistances) and different power losses to the environment.
- At *cold start*, the printer has to be heated up initially before the first sheet can be printed. During a cold start the printer will run, but without sheets, to get equally heated components and belts inside the printer. Without paper, the parameters of the printer (thermal capacities and resistances) have considerably different values.
- The *environment temperature* determines the thermal power loss of the printer. It is not measured, but any controller has to cope with its influence. Although unknown, the environment temperature will not change quickly. A controller has to compensate its influence.

- An even slower process is *wear*. At a longer time scale it will change the process parameters owing to aging, pollution and degrading performance of subsystems. A controller has to take care of slowly changing parameters and counteract them.

Next we discuss the control inputs u. The inputs $u = (P_{\text{pre}}, P_{\text{TTF}}, v, d)$ can be manipulated freely to influence the system. These input are generated by a controller to compensate all disturbances mentioned, such that print quality is not sacrificed and print throughput is still at a maximum. The measured outputs y include the preheating temperature T_{pre} and the TTF temperature T_{TTF}. That is $y = (T_{\text{pre}}, T_{\text{TTF}})$. With reference to the heat model (4.6), the system states x are defined in a 4-dimensional vector

$$x := \begin{bmatrix} T_{\text{pre}} \\ T_{\text{TTF}} \\ T_{\text{roller}} \\ T_{\text{slow}} \end{bmatrix}.$$

- Both the preheater and the TTF belt can be heated with separate electrical powers (P_{pre} and P_{TTF}). The maximum current of a power connection is specified, while the voltage, and so the power, can change. However, both have a maximum and together they have to satisfy the maximum amount of electrical power for heating. The maximum can change owing to other, higher priority jobs inside the printer.
- When heavy paper sheets are being printed, it requires more thermal and so more electrical power to heat these sheets to the required temperature. When the constraint on electrical power is active, the same temperatures can still be achieved by either decreasing the print speed v or by increasing the distance d between the sheets.

The following constraints have to be satisfied.

- Input constraints: power, velocity, and distance are considered variable according to

$$P_{\text{pre}}(t) + P_{\text{TTF}}(t) \leq P_{\max}(t), \tag{4.8}$$

$$|\Delta v(t)| \leq \bar{v}, \tag{4.9}$$

$$v_{\min} \leq v \leq v_{\max}, \tag{4.10}$$

$$d_{\min} \leq d \leq d_{\max}. \tag{4.11}$$

  where $\Delta v(t)$ is the change rate of velocity.
- Print process constraints: tight bounds for the fuse temperature to guarantee a minimum print quality.

The goal will be to maximise throughput Λ [ppm] while satisfying print quality (fuse temperature constraints) and all technical constraints, such as the maximum available electrical power. Throughput Λ depends on the velocity v of the printer and on the sheet length l and the distance d between two sheets:

$$\Lambda = \frac{60v}{l+d}. \tag{4.12}$$

To achieve that goal, one consistent and transparent problem formulation with all aspects operating a printer has to be considered. Solving this large optimisation problem will result in the optimal compromise between all conflicting requirements.

Looking at the characteristics of the disturbances it can be noted that a printer changes its behaviour considerably during printing. Especially the fast changes in requested paper sheets, introduce large changes in the parameters of the printer. Consequently, one optimal controller for all operating conditions cannot be expected to deliver the required print quality. The large changes demand a robust or an adaptive controller. The slowly varying environment temperature and the consequences of wear can be taken into account by many controllers.

A high throughput is required. Consequently, at some point in time the operation will hit one or more constraints, being the electrical power for heating, the velocity, or the fuse temperature.

In spite of the many changing parameters, the basic model describing the temperatures in the printer remains valid, although several parameter values will change. A deterministic control approach, with extra features to follow the changing operating conditions of the process seems to be a good choice for this problem. A stochastic approach, for example Gaussian processes [14, 15], which neglects the, known, deterministic relations among all variables, has to retrieve all information based on statistical and stochastic properties of the measured variables. An example of the output of such a Gaussian process is shown in Fig. 4.5. Besides an expected value, also the (growing) uncertainty is calculated for increasing prediction horizons. This requires more considerable measurement and calculation time in real-time applications. It can certainly cope with changes in the environment temperature and wear, but it will be too slow to anticipate or counteract the fast changes introduced by printing on different sheets.

Within the class of deterministic control approaches, robust and adaptive control techniques can efficiently cope with time-varying behaviour of the printing system. However, as the largest disturbance is known (print jobs can be predicted), adaptive control is to be preferred over robust control. Robust control without adaptation will be too slow and consequently cannot anticipate fast enough on the known disturbances.

This brings us to the conclusion that adaptive control is most likely the preferred control approach for the control problems formulated for the industrial electro-photography printer. Moreover, any adaptive method that can utilise predicted values of the disturbance and can deal with the many hard constraints is to be preferred.

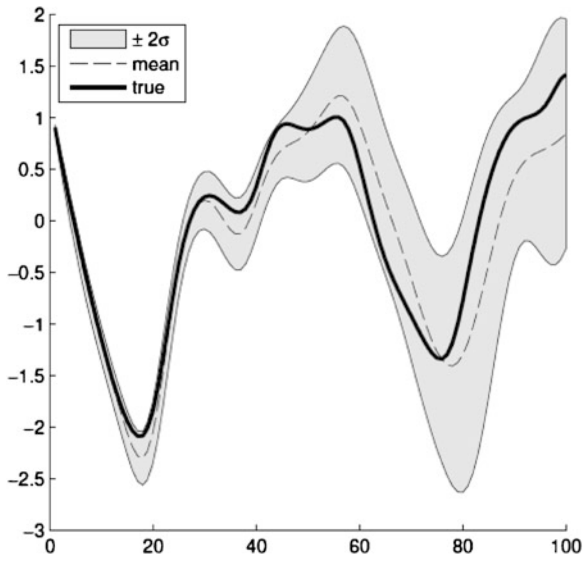


Fig. 4.5 Increasing uncertainty when predicting with a Gaussian process

4.3 Control Design

The major issues that are encountered in the printing system are related to large and fast parameter variations and disturbances (paper size, mass, and humidity) and constraints. These problems influence the performance of the printing system. Good quality means that the fusing temperature should be at a certain desired (print) job-dependent temperature level for all different print jobs. Currently, an industrial PI controller is implemented in the printing system to control the preheating and TTF temperatures. The advantages of the PI controller include a simple structure, easy to design and implement, and there is no need for an accurate model of the process. However, a PI controller has some difficulties in the presence of constraints, it does not adapt itself to changing process behaviour, it is slow in responding to large disturbances, and it does not guarantee optimal performance. In this section, we present the application of three different suitable control strategies to handle these issues.

- *Model Reference Adaptive Control (MRAC)*. MRAC adapts the controller parameters to the operating condition of the process, without explicitly estimating the model or parameters of the process. The same signal is inserted to both the input u of the reference model and to the reference r of the controlled process. Then both outputs are compared which yields the error e. The adaptive mechanism tries to reduce this difference e to zero by adjusting the controller, as presented in Fig. 4.6. When this error is zero, there is no difference between the reaction of the reference model and of the controlled system. So the controlled real process has

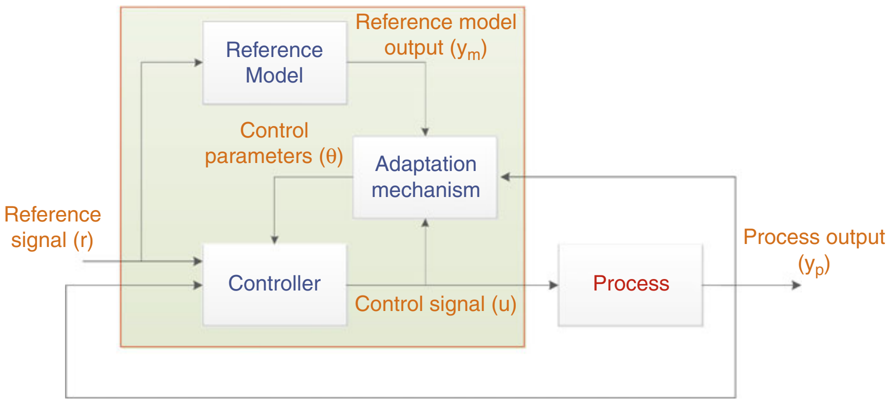


Fig. 4.6 Main structure of Model Reference Adaptive Control (MRAC)

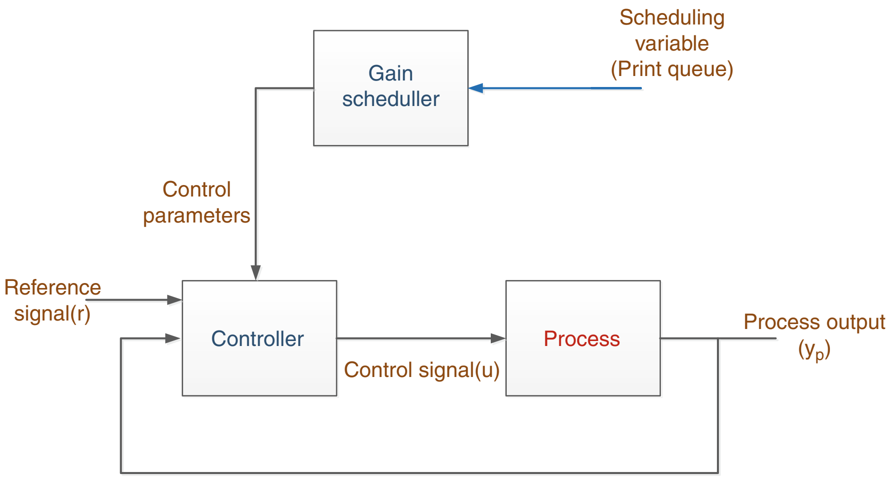


Fig. 4.7 Main structure of gain scheduling robust control

the same behaviour as the selected reference model in spite of disturbances and changing parameters. It is a relative easy control approach with nice performance.

- *Gain scheduling robust control*. The largest disturbance, the print queue, is known in advance. So, an appropriate, pre-designed controller can be calculated for each possible paper sheet in the print queue. When the paper sheet is known, the appropriate controller is taken from the list of available controllers and is applied, see Fig. 4.7.
- *Model Predictive Control (MPC)*. In contrast with the previous methods, MPC uses a prediction horizon. Based on a deterministic model and a to-be-selected input, this technique amounts to predicting the process output. When a criterion

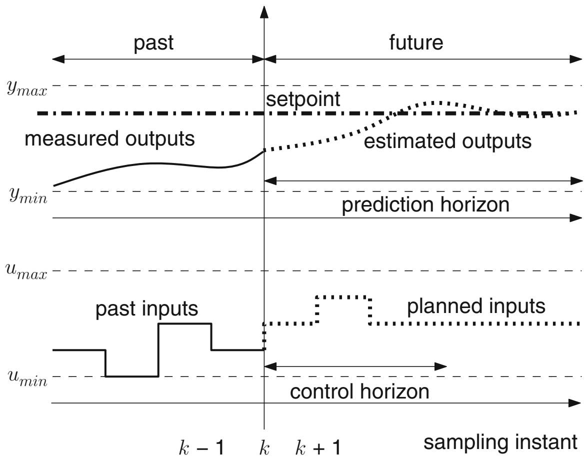


Fig. 4.8 Main structure of model predictive control (MPC)

is utilised that expresses the requirements, even an optimal input sequence can be calculated. As long as the output can be calculated, even when constraints are introduced or known disturbances, still the optimum input sequence can be found. So, MPC can anticipate future sheet characteristics in the print queue and it can take hard constraints explicitly into account. Typically, in MPC the control horizon, the number of implemented control samples, is shorter than the prediction horizon, the number of estimated output samples. The basic idea of MPC is depicted in Fig. 4.8. The lower plot represents the selected inputs (dotted line), while the upper plot represents the calculated output based on these selected inputs. The inputs are selected such that a cost function is minimised over the receding prediction horizon N. A typical cost function assumes the form

$$\mathscr{J}_k(u) = \sum_{i=k+1}^{k+N} \Big((y_{\text{ref}}(i) - y(i))^\top Q_i (y_{\text{ref}}(i) - y(i)) + u(i)^\top R_i u(i) \Big),$$

where y_{ref} denotes the reference variable (e.g. desired temperature), y is the controlled variable (e.g. measured preheating and TTF temperatures), k represents the time instant, Q_i weighting coefficient reflecting the relative importance of the output y, and R_i is a weighting coefficient penalising relative big changes in the input u.

Each control strategy will be shortly described with emphasis on its ability to adapt itself to the disturbances and parameter changes introduced by the different paper sheets in the print queue. The theory behind these methods is explained in the references. After the discussion of these three control approaches, several examples show the characteristics of these controllers for the professional printer example.

4.3.1 Model Reference Adaptive Control

An adaptive controller modifies a control law so as to cope with the time-varying or uncertain parameters of the system being controlled [1, 3]. MRAC is one of the approaches for adaptive control. Model Reference Adaptive Control (MRAC) was first introduced by Whitacker in 1958. Over the past decades, various MRAC methods have been investigated. The majority of MRAC methods may be classified as direct, indirect, or a combination of these. Indirect adaptive control methods are based on the identification of unknown plant parameters and define control schemes derived from the parameter estimates. Parameter identification techniques such as recursive least-squares and neural networks have been used in indirect adaptive control methods. On the other hand, direct adaptive control methods directly adjust control parameters to account for system uncertainties without identifying unknown plant parameters explicitly. Since the printing system has a fast-varying parameters, the adaptation phase has to be as short as possible. Therefore, in this section we focus on direct adaptive control.

The basic structure of a direct MRAC scheme is shown in Fig. 4.6. MRAC schemes aim to reduce the tracking error $e_1 = y_p - y_m$ between the plant output y_p and the output of the reference model y_m. A stable reference model is designed to achieve a desired control performance. The closed-loop system consists of an ordinary control configuration with a feedback control law that contains the plant and a controller $C(\theta)$ and an adjustment mechanism that optimally adjusts the controller parameters $\theta(t)$ in real-time to force the controlled plant to follow the reference-model output. The design procedure involves the use of a wide class of adaptive laws that include least-squares, gradient, and SPR-Lyapunov design approaches [25, 28].

Adaptive control has been successfully implemented in several applications [1,3]. However, the possible high-gain control for fast adaptation is an issue. In some applications, like the professional electro-photography printers, fast adaptation is required to improve tracking performance when a system is subject to large uncertainties. In this case, a large adaptation gain must be used to reduce the tracking error rapidly. However, there typically exists a balance between stability and speed of adaptation. A fast adaptation gain results in high-frequency oscillations, which can excite unmodelled dynamics that could adversely affect the stability of an MRAC system. On the other hand, small adaptation gains will result in an unacceptably slow response.

In this section, we address two different methods to improve the convergence of MRAC, namely, using a non-linearly varying adaptation gain and using multiple adaptation gains with a new adaptation law.

In [8], we have proposed a non-linear time-varying adaptation gain. The design does not require any knowledge of the parameters of the system. To improve the system performance, the adaptation gain should be chosen as a function of the controller parameters error. However, the optimal controller parameters depend on the process parameters, which are usually unknown. The error between the outputs of the process and of the reference model gives a useful indication of the controller parameter errors. Hence, we propose an adaptation gain as a function of the output error instead of controller parameter error.

Besides state feedback, we have also derived adaptation laws on the basis of output feedback in [7], but those will not be discussed here.

Consider a linear time-varying system given by

$$\begin{aligned}\dot{x}(t) &= A_p(t)x(t) + B_p(t)u(t),\\ y(t) &= C_p x(t),\end{aligned} \tag{4.13}$$

where $x(t)$ is the state vector of the plant (professional printer), $y(t)$ is the plant output, and $u(t)$ denotes the plant input. For the professional printer, these variables are the ones that are defined in Sect. 4.2.2. The matrices $A_p(t)$, $B_p(t)$, C_p are the state space realisation of the printing system model (4.6), which are assumed to be time varying.

Suppose that the stable reference model is given by

$$\begin{aligned}\dot{x}_m(t) &= A_m x_m(t) + B_m r(t),\\ y_m(t) &= C_m x_m(t),\end{aligned} \tag{4.14}$$

where $r(t)$ is a reference input signal, $x_m(t)$ is the state vector of the reference model, $y_m(t)$ is the reference output, A_m, B_m, and C_m are the state space matrices of the reference system. The reference model is chosen to represent a desired performance of the closed-loop system.

It can be shown that with the state feedback control law

$$u(t) = K_{\mathrm{FF}}(t)r(t) + K_{\mathrm{FB}}^{\top}(t)x(t), \tag{4.15}$$

which is written in a compact form as

$$u(t) = \theta^{\top}(t)V(t), \tag{4.16}$$

where $\theta(t) = col(K_{\mathrm{FF}}(t),\ K_{\mathrm{FB}}(t)) \in \mathbb{R}^{1+n}$ denotes the controller parameter vector, n represents the number of states, and $V(t) = col(r(t),\ x(t)) \in \mathbb{R}^{n+1}$ is the regressor vector.

An adaptive control law exists that defines the evolution of $\theta(t)$ and that forces the difference $e(t) = x(t) - x_m(t)$ to zero. So ultimately the state of the process $x(t)$ and the state of the reference model $x_m(t)$ will become the same. The process behaves like the (desired) reference model in spite of disturbances and parameter uncertainty. For time-invariant models (4.13) and (4.14), the closed-loop system is asymptotically stable ($e \to 0$ as $t \to \infty$) using the control law (4.16) and the update law

$$\dot{\tilde{\theta}} = \dot{\theta} = -\Gamma w B_I^{\top} W e, \tag{4.17}$$

with $B_I = [0 \quad \cdots \quad 0 \quad 1]^\top \in \mathbb{R}^n$ and $W \succ 0$ a positive definite matrix, which satisfies $A_m^\top W + WA_m \prec 0$. The speed of convergence is determined by a gain Γ. This adaptation gain Γ determines the rate at which the controller parameter will converge to steady-state values. Moreover, the adaptation gain influences the performance of the system. Hence, the adaptation gain should be properly chosen. A high adaptation gain may lead to badly damped behaviour, while a low adaptation gain will lead to an unacceptable slow response.

A new MRAC state feedback design is introduced based on a non-linear adaptation gain. The adaptation gain Γ is chosen as a function of the error e such that if the error is large, the adaptation gain Γ will be large. This implies that the controller will adapt its parameters faster and thus a faster convergence of the error is achieved.

Let the adaptation gain be

$$\Gamma = \gamma_0 + \gamma_1 e^\top Pe, \quad \gamma_0 > 0, \quad \gamma_1 > 0, \quad P \succ 0, \quad \text{therefore} \quad \Gamma > 0. \tag{4.18}$$

Using the state feedback control law (4.16), the adaptation law (4.17), and the non-linear adaptation gain (4.18), the closed-loop system is still asymptotically stable. Note that γ_0 and γ_1 are design parameters. The non-linear adaptation gain (4.18) is more generic than the standard MRAC with a constant adaptation gain ($\gamma_1 = 0$). It will show faster convergence without sacrificing stability.

A further performance improvement of MRAC is obtained in [10], where a novel adaptation law is introduced for both state and output feedback. Unlike the standard MRAC, which uses a single constant adaptation gain, multiple adaptation gains are employed in this approach. The error dynamics, which are composed of the output tracking error and the controller parameter estimation error, are first represented by a Takagi-Sugeno (T-S) model [26]. The T-S model is considered as an exponentially stable system perturbed by an external disturbance. Therefore, the adaptive control problem is formulated as minimising the $\mathscr{L}_2$ gain. By this formulation, the optimal adaptation gains are obtained by solving a linear matrix inequality (LMI) problem [2].

In this approach, a new adaptation law is defined as

$$\dot{\theta}(t) = \sum_{i=1}^{L} h_i(V(t))\,(\Gamma_i e(t) + \beta_i \theta(t)) \tag{4.19}$$

where, for all i, $h_i : \mathbb{R}^{n+1} \to \mathbb{R}$ is such that $h_i(V(t)) \geq 0$ and $\sum_{i=1}^{L} h_i(V(t)) = 1$ for all $V(t) \in \mathbb{R}^{n+1}$. Here, L represents the number of subsystems and Γ_i and β_i are adaptation design variables.

Using the state feedback control law (4.16) with the adaptation law (4.19), an LMI feasibility problem is formulated such that the error dynamics of the closed-loop system is guaranteed to be asymptotically stable.

The key advantage of this approach is to select in real-time the best adaptation gain from an a priori designed set of adaptation gains. The adaptation gain is selected based on the operating point of the system states. Consequently, the adaptation transients have been considerably improved. For more details about this approach, interested readers are referred to [7].

4.3.2 *Gain Scheduling Robust Control*

The Takagi-Sugeno (T-S) model has been widely used in various applications since it can efficiently model and control complex non-linear systems. A T-S model, which was introduced in [26], is composed of a weighted sum of local linear models. A T-S model approximates a non-linear system and the weights in the T-S model depend on the operating point of the non-linear system. According to the universal approximation theorem [29], a T-S model can approximate a non-linear system arbitrarily well. Recently, T-S model-based controllers have been applied to stabilise non-linear systems [4, 27].

The T-S model is formally presented as a weighted average of several linear models through weighting functions. The T-S model is defined as:

$$\Sigma : \begin{cases} \dot{x}(t) = \sum_{i=1}^{L} h_i(z(t))\,(A_i\,x(t) + B_i\,u(t))\,, \\ y(t) = \sum_{i=1}^{L} h_i(z(t))\,C_i\,x(t), \end{cases} \tag{4.20}$$

for $i = 1,\ldots,L$, where $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{n\times m}$, $C_i \in \mathbb{R}^{p\times n}$, and L denotes the number of linear models. Note that all matrices have identical dimensions independent of their index i. $h_i(z(t)) \geq 0$ for $i = 1,\ldots,L$, $\sum_{i=1}^{L} h_i(z(t)) = 1$, and $z(t)$ is a triggering (scheduling) variable, which is usually dependent on the current value of the system state $x(t)$. $z(t) \in \mathbb{R}^n$ is the variable that decides about the operating point. In our professional electro-photography printer, $z(t)$ represents the known print queue.

The structure of the T-S model has led to the design of a feedback controller for each local model, which are further combined into a single overall controller by a weighted combination as follows

$$u(t) = \sum_{j=1}^{L} h_j(z(t))K_j x(t). \tag{4.21}$$

Here, K_j are the controller gains for $j = 1,\ldots,L$ and the weighting functions $h_j(z(t))$, for $j = 1,\ldots,L$, are the same as in (4.20).

The T-S model structure has been used as a feasible approach to capture the dynamic characteristics of printing system under different operating conditions. The approximation error (or misfit) between the original non-linear printing system and the T-S model is known as the *consequence uncertainty*. Designing a stabilising controller based on the T-S model may not guarantee the stability of the original non-linear system under such a controller.

In [9, 11], we have addressed the robust control problem of an electro-photography printing system, using an $\mathscr{L}_2$ state and output feedback controller. The non-linear printing system is approximated by a Takagi-Sugeno (T-S) model. A robust control technique is proposed to cope with the effect of the approximation error between the non-linear model of the printing system and the approximating T-S model. A sufficient condition is derived to ensure robust stability of an $\mathscr{L}_2$ state and output feedback controller with a guaranteed disturbance attenuation level. A technique based on a parameterised Lyapunov function is employed in our approach. The control problem is formulated in terms of a linear matrix inequality for which efficient optimisation solvers are used to test feasibility. In this framework, we utilise the known paper characteristics of the print queue as triggering (scheduling) variable. So, for each paper mass we can select the best fitted controller. The implementation of this technique for the printing system is explained in more detail in Sect. 4.4.

4.3.3 Model Predictive Control

An important research challenge is to design a run-time adaptive system that maintains a high level of economic performance, i.e. to maximise throughput for the printer, under non-linear dynamic behaviour and changing, partly known operating conditions. To deliver high performance, the system should adapt itself in run-time, based on the operating conditions and constraints. In our case, the adaptation is mainly related to changes in the media type (print queue). Model predictive control (MPC) is a good choice to control the behaviour of printers, because it can deal in an optimal way with hard constraints and available information of the print queue [21]. MPC makes use of a model of the system to compute the optimal inputs. Since the paper properties directly influence the system dynamics, different models should be used in control for different paper types, when heating up the system and printing. A non-linear parameter-varying model of the system could capture all the dynamics of the process. Several approaches have been proposed in literature to deal with model predictive control for linear parameter-varying systems e.g. [19, 31]. When large transitions between setpoints are required, as in the case of changing operating points, tracking control of a constrained non-linear system is a challenging problem [12, 22].

Systems with input-induced non-linearities [5] are a special type of non-linear systems. They are a class of input-dependent non-linear systems. Some of the inputs are considered to be more active in the dynamic behaviour of the system than others. If inputs and known disturbances that induce non-linear behaviour are constant for some period, the system can be approximated by a linear one in those periods. Therefore, the input is divided into two components $u(k) = col(u_d(k), u_l(k))$, where

$u_d := col(v,d)$ and $u_l := col(P_{\text{pre}}, P_{\text{TTF}})$. According to the printing system heat model (4.6), with u_d constant, the system will be linear with respect to u_l.

A good control approach, which assures both maximum throughput and the high quality of the system, is based on a non-linear dynamic input-dependent predictive controller. It will be presented in this section. The controller has to track a variable target, while respecting hard constraints of the plant, coping with the uncertainties due to modelling errors and partly known disturbances. Based on the printing system model, we formulate the control problem into centralised and decomposed strategies.

4.3.3.1 Centralised Control Strategy

The objective is to design a controller which can assure maximum throughput while keeping the print quality within constraints under all operating conditions. Knowing a few seconds in advance what type of paper will come in the fusing point, the question is how to determine the optimal heating strategy for the printing process in cold start and warm process. This information can be used in feedforward to obtain a better performance of the system. The printer needs to be heated up to a certain temperature to make sure that the print quality is not lost, not even for one page. During printing, there may be several switches between different types of paper. Optimising the distribution between P_{pre} and P_{TTF}, v and d, can improve the throughput during switches.

The control scheme will have to adapt itself between several modes, e.g. cold start, warm process, depending on the media type.

Given a non-linear system represented by a discrete-time model:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), \theta(k), w(k)) \\ y(k) &= g(x(k), u(k), \theta(k)) \end{aligned} \tag{4.22}$$

where x is the system state vector, u is the input vector, $\theta(k)$ is the time-varying input-dependent known parameter vector, which represents the known paper properties, w is the unknown disturbance vector, and y is the measured output of the system. The system is subject to hard constraints on x, u, y, and w, as explained in Sect. 4.2.2. The output of the system y contains, beside the measured outputs (T_{pre}, T_{TTF}), the estimated fuse temperature T_{fuse}.

In industrial control systems, the goal is to optimise dynamic plants from an economic point of view. It is required to steer the system along a time-varying target. The controller has to track the target, given the hard constraints of the plant, while considering the uncertainties included in system the modelling and measured and unmeasured disturbances. The time-varying target can be determined based on real-life observations or based on optimisation rules, given some specific criterion.

Since most of the industrial plants are subject to hard constraints, a good solution of the problem can be based on a model predictive formulation.

The model predictive control (MPC) method is conceptually a method for generating feedback control actions for linear and non-linear plants subject to constraints, especially if predictions of the disturbances are available (θ). It is one of the most successful control techniques in process industry [21]. In principle, based on the past inputs and outputs, but also on the future control scenario (the control actions that we intend to apply from the present moment k onwards), at each sample k, the process output y is predicted over a prediction horizon N. A reference trajectory is defined over the prediction horizon, to force the process output to follow a predefined trajectory y_{ref}. This information is used by the controller to provide the optimal control input vector u^*, according to a predefined optimisation cost function. Only the first element of the optimisation will be applied to the process, and the remaining elements of the optimisation will be discarded. The same procedure will be repeated at each sample.

The optimal solution $u^* = col(u_d^*\ u_l^*)$, at time instant k, satisfies:

$$u^* = \arg\min_{u} \sum_{i=1}^{N} [\mathscr{J}(u_d(i), \theta(i)) + \lambda (y(i) - y_{\mathrm{ref}}(u_d(i), \theta(i)))^2] \tag{4.23}$$

such that

$$\begin{aligned}
x(0) &= \hat{x}(k) \\
x(i+1) &= f(x(i), u(i), \hat{w}(i), \theta(i)) \\
y(i) &= g(x(i), u(i), \theta(i)) \\
i &\in I_0^{N-1} = \{0, \ldots, N-1\} \\
x(i) &\in \mathrm{X} \quad u(i) \in \mathrm{U} \quad y(i) \in \mathrm{Y} \\
\sum_{j=1}^{m_l} u_{l_j}(i) &\leq u_{\max}(k) \\
|u_d(i) - u_d(i-1)| &\leq \overline{u_d}.
\end{aligned}$$

A trade-off is being made between the economic performance of the system and output tracking in the cost function using the penalty function λ. The economic performance, e.g. printers with high throughput are desired, is described by $\mathscr{J}(u_d(i), \theta(i))$ (4.12). The output of the system $y(i)$ is forced to follow a desired value $y_{\mathrm{ref}}(u_d(i), \theta(i)))$ (e.g. required performance criterion for print quality computed based on knowledge of known input u_d and estimated θ) for each time instant, $i \in I_0^{N-1}$, with N the prediction horizon. For λ small, output tracking does not play an important part in the cost function, while for increased values of λ output tracking becomes dominant. Over the prediction horizon N, the values of the system parameters $\theta(i)$ are known and they can change each sample, $\theta(i) \in \Gamma$. The estimated values of disturbances are between bounds $\hat{w} \in \Phi$. The non-linear optimisation is performed in the presence of hard constraints. The sum of u_l inputs

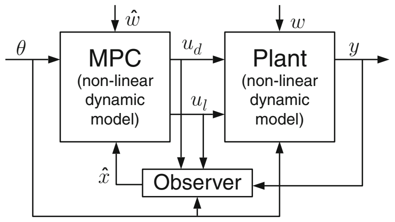


Fig. 4.9 Centralised control scheme of the system

is limited any moment in time by a time-varying maximum $u_{\max}$ (e.g. a physical limitation of the available power). The variation in $u_{\max}$ is random (no model available) but bounded. Since no prediction can be made for the future, $u_{\max}$ is assumed to be constant and equal to the last known value. Due to some physical limitations of the system, an upper bound $\overline{u_d}$ of the rate of change in u_d input is taken into account in the problem formulation. A piece-wise linear observer is built to estimate the system states $\hat{x}$, since not all the states are measurable. It uses a linearised system model around $u_d(k)$ computed in the previous step and known $\theta(k)$.

The control scheme of the process, including the controller and the observer, is shown in Fig. 4.9.

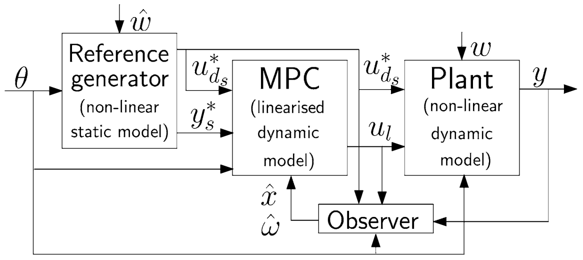
To assess the stability of the system, a control Lyapunov function (CLF) can be derived and expressed in the optimisation problem [18, 24].

As it was mentioned in Sect. 4.2.2, an interesting challenge for printers is the cold start. To maximise the productivity of the printer, it is important to determine the optimal heating strategy when the printing process can start. The machine needs to be heated up as fast as possible to achieve print quality for the first page too. After the printing process starts, the speed and the distance between sheets need to be adjusted, until a steady state is reached. The process is highly non-linear and influenced by several disturbances during cold start. All the dynamics are changing, and in case of shortage in power, the power distribution plays an important role in determining the system performance.

For the given input-dependent model of the system (4.22), the switching between printing modes (cold start, warm process), translates into adaptation of the control scheme to a new set of parameters θ, e.g. cold start without paper $\theta = \theta_1$, warm process with different types of paper $\theta = \theta_i$, with the index i represent the type of paper.

4.3.3.2 Decomposed Control Strategy

The printing system (4.6) is represented by an input-induced non-linear dynamic model, as explained in Sect. 4.3.3. For the centralised formulation, all four inputs are computed dynamically using the non-linear dynamic model of the printing system. However, here the non-linear dynamics of the system could be divided into



$\hat{w}$
Reference generator (non-linear static model)
θ
$u_{d_s}^*$
y_s^*
MPC (linearised dynamic model)
$u_{d_s}^*$
u_l
w
Plant (non-linear dynamic model)
y
$\hat{x}$
$\hat{\omega}$
Observer

Fig. 4.10 Decomposed control scheme of the system

a non-linear static model and a linearised dynamic model. Therefore, the control system can be decomposed into two levels to apply a second control approach, as shown in Fig. 4.10.

- *Reference generator*. This is a non-linear feedforward optimisation scheme used in run-time for reference generation. It uses the static non-linear model of the system and takes into account the operating conditions, the system constraints and known parameters for N samples. This static optimisation assures maximum throughput in steady-state. The optimal steady-state solution for $(u_{d_s}^*, u_{l_s}^*, y_s^*)$ is determined. The used scheme decouples the u_d input, which makes the printing system linear for the feedback control loop (MPC level).
- *MPC*. In this level, setpoint tracking and print quality satisfaction (robustness against modelling errors and unmeasured disturbances) are achieved using a linear model predictive controller (MPC) formulation. The design of the MPC employs a linearised model of the system around the optimal $u_{d_s}^*$ obtained from the static optimisation level. To obtain off-set free tracking with the MPC, i.e. zero steady-state error, a disturbance filter is used [21], to estimate and predict the mismatch between measured and predicted outputs. An observer is designed to estimate the system state $\hat{x}$ and the disturbances $\hat{\omega}$.

This control strategy requires less computation time compared to the control scheme described in Sect. 4.3.3.1. That makes it attractive for real-time implementation for the printing system. However, transient performance cannot be guaranteed when changing the operating conditions because part of the inputs are obtained based on static optimisation.

4.4 Professional Electro-Photography Printing Systems under Control

In this section, we present the implementation of the three control schemes described in Sect. 4.3 to the printing system. We also compare between the performance of both these control schemes and the industrial PI controller that is current used in the printer. To have a fair comparison, the PI controller is a well-tuned industrial controller.

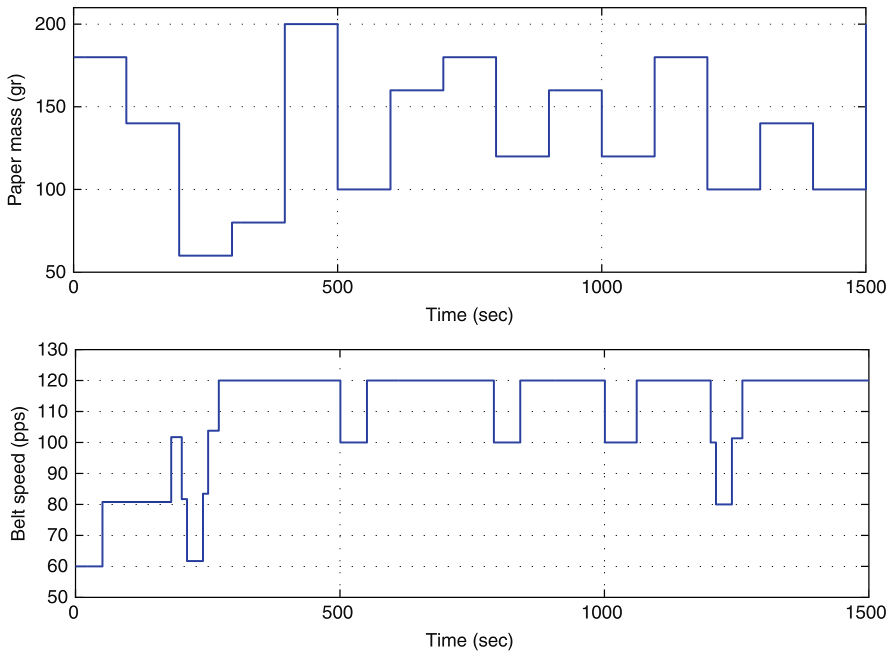


Fig. 4.11 The variation of the paper mass and the transfer belt speed

4.4.1 Model Reference Adaptive Control

Based on the printer characteristics, MRAC is suggested to control the fuse temperature. The objective is to design a controller to keep both the print quality and the productivity as good as possible. Here, we present a comparison of the tracking control results of the existing industrial PI controller and the proposed MRAC approaches.

In the printing system (4.6), the control input is $u(t) = col(P_{\text{pre}}(t), P_{\text{TTF}}(t))$, and the measured output is $y(t) = col(T_{\text{pre}}(t), T_{\text{fuse}}(t))$. Several parameters of the printing system are time-varying since these parameters depend on the different printing jobs, while other parameters are unknown. As shown in Fig. 4.11 the paper mass and the speed are varied to simulate the parameter variations.

To achieve a short transient time with little overshoot, we choose the reference models for the preheater and the TTF as

$$A_{r_{\text{pre}}} = -0.5, \quad B_{r_{\text{pre}}} = 0.5, \quad C_{r_{\text{pre}}} = 1,$$

$$A_{r_{\text{TTF}}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.125 & -0.75 & -1.5 \end{bmatrix}, \quad B_{r_{\text{pre}}} = \begin{bmatrix} 0 \\ 0 \\ 0.125 \end{bmatrix}, \quad C_{r_{\text{pre}}} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

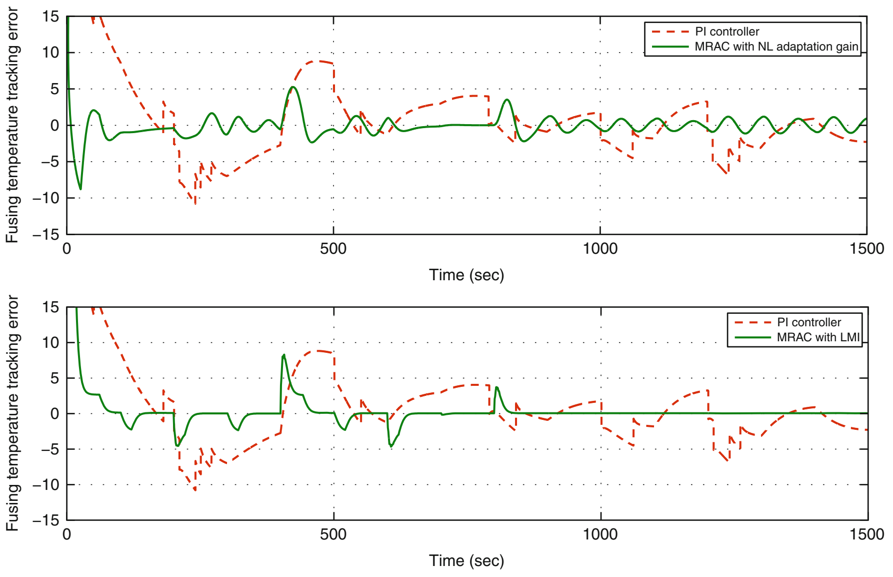


Fig. 4.12 Fusing temperature tracking error

The designer is free to choose the reference model according to the desired closed-loop performance.

Since the fusing temperature depends on the preheating and the TTF temperature, we apply MRAC with the non-linear adaptation gain to control both the preheating and TTF temperature. We also implement the adaptive control law with the multiple adaptation law. To construct the T-S model that approximates the error dynamics, we assume that $V_{\text{pre}} \in [\mathscr{N}_{\min}, \mathscr{N}_{\max}]$ and $V_{\text{TTF}} \in [\mathscr{M}_{\min}, \mathscr{M}_{\max}]$. We choose

$$\mathscr{N}_{\min} = col(50, 50), \quad \mathscr{N}_{\max} = col(100, 100),$$

$$\mathscr{M}_{\min} = col(0, 0, 0, 0), \quad \mathscr{M}_{\max} = col(100, 100, 1000, 1000).$$

The triangle shape weighting functions are adopted for this approach.

Figure 4.12 compares the fusing temperature tracking error of the existing industrial PI controller, the MRAC with a non-linear adaptation gain, and the MRAC with multiple adaptation gains. As depicted, the tracking performance of both MRAC schemes are better than that of the PI controller in the presence of large parameter variations. Figure 4.13 presents the preheating temperature tracking of different temperature levels with paper mass variations. These simulation results illustrate that the proposed MRAC schemes can efficiently improve the performance of the printing system in the presence of large parameter variations.

Using MRAC with a non-linear adaptation gain guarantees a faster convergence of the tracking error to zero but it does not guarantee the convergence of the controller parameters to the optimal value. On the other hand, employing the adaptive

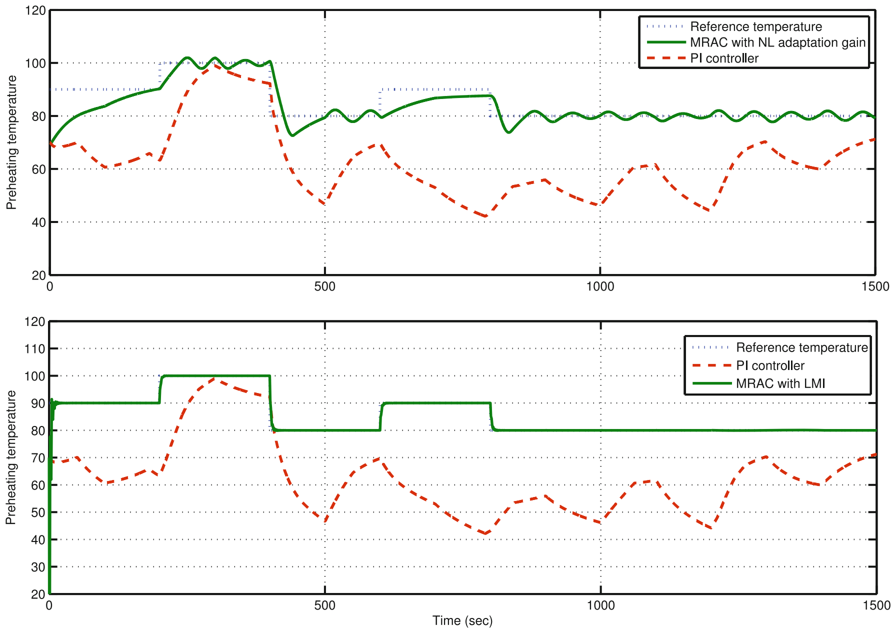


Fig. 4.13 Preheating temperature tracking control comparison

law with multiple adaptation gains yields convergence of both the tracking and the controller parameters errors. Thanks to the representation of the augmented error as a T-S model, the adaptive problem is formulated as an LMI feasibility problem and, therefore, the optimal adaptation gains are obtained. That explains the better performance of the MRAC with multiple adaptation gains compared to the MRAC with the non-linear adaptation gain.

4.4.2 Gain Scheduling Robust Control

In this section, we present the implementation of the robust control scheme to the electro-photography printing system and the comparison between the performance of both the proposed robust control scheme and the existing industrial PI controller.

Given the knowledge of the paper mass m and the belt speed v, the printing system dynamics are approximated with a T-S model. Note that the choice of the number of the sub-models L depends on the available knowledge about the operating conditions of the system. A large number of sub-models will result in a small approximation error with high complexity and vice versa. There is always a trade-off between complexity and accuracy. Therefore, the T-S model which approximates the printing system (4.6) is represented using the following set of nine linearised models:

$$\dot{x}(t) = \sum_{i=1}^{9} h_i(z(t)) \left(A_i x(t) + B_i u(t) + E\hat{d}(t)\right),$$
$$y(t) = \sum_{i=1}^{9} h_i(z(t)) C_i x(t). \tag{4.24}$$

We choose the following nine working regions R*i* to describe the dynamics of the printing system:

R1: $m \leq 80$ and $v \leq 60$,
R2: $m \leq 80$ and $60 \leq v \leq 90$,
R3: $m \leq 80$ and $90 \leq v \leq 120$,
R4: $80 \leq m \leq 140$ and $v \leq 60$,
R5: $80 \leq m \leq 140$ and $60 \leq v \leq 90$,
R6: $80 \leq m \leq 140$ and $90 \leq v \leq 120$,
R7: $140 \leq m \leq 200$ and $v \leq 60$,
R8: $140 \leq m \leq 200$ and $60 \leq v \leq 90$,
R9: $140 \leq m \leq 200$ and $90 \leq v \leq 120$,

where $x(t) = \begin{bmatrix} T_{\text{pre}} & T_{\text{TTF}} & T_{\text{roller}} & T_{\text{slow}} \end{bmatrix}^\top$, $u(t) = \begin{bmatrix} P_{\text{pre}}(t) \\ P_{\text{TTF}}(t) \end{bmatrix}$, $\hat{d}(t) = T_{\text{env}}(t)$ is the environment temperature, $y(t) = \begin{bmatrix} T_{\text{pre}}(t) \\ T_{\text{TTF}}(t) \end{bmatrix}$, and the triggering variables $z(t) := col(m, v)$.

After several iterations using the LMI optimisation toolbox in MATLAB [23], we found the optimal controller parameters that guarantee an $\mathscr{L}_2$ gain is 0.5. As shown in Fig. 4.14, the external disturbance $\hat{d}(t)$ (environment temperature) is assumed to vary between 15 and 30 °C. The variation of the paper mass and the belt speed is depicted in Fig. 4.11. To achieve a good print quality, the T_{TTF} and T_{pre} should be kept at a certain desired setpoint. T_{TTF} and T_{pre} are used to estimate the fusing temperature. Figure 4.15 shows the fusing temperature tracking error comparison. As shown, the tracking performance of the robust controller is better than the PI controller in the presence of the parameter variations and the external disturbance. Figure 4.16 shows the preheating temperature tracking of different temperature levels with paper mass variations. The simulation results show that the observer-based $\mathscr{L}_2$ controller has considerably improved the print quality with relatively large external disturbances while the desired performance is still being achieved.

4.4.3 Model Predictive Control

In this section, the control performance will be analysed of both the centralised and the decomposed MPC controllers. These controllers must be robust with respect to

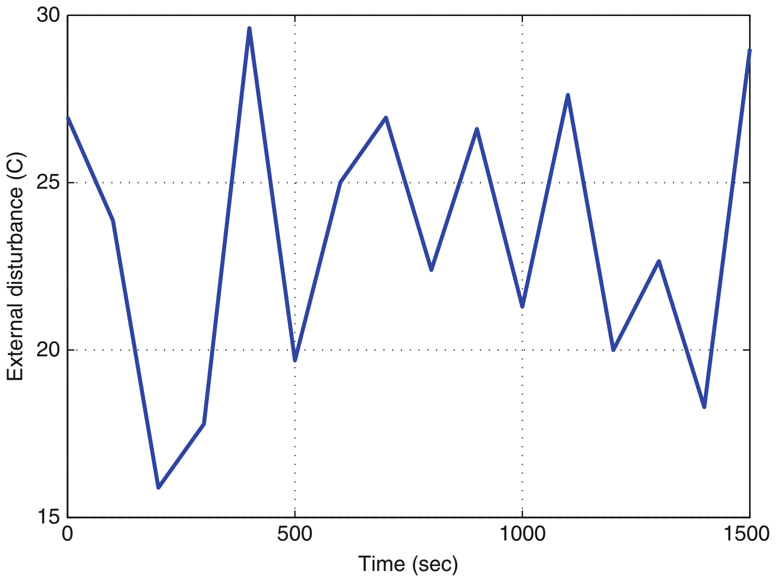


Fig. 4.14 The external disturbance $\hat{d}(t)$

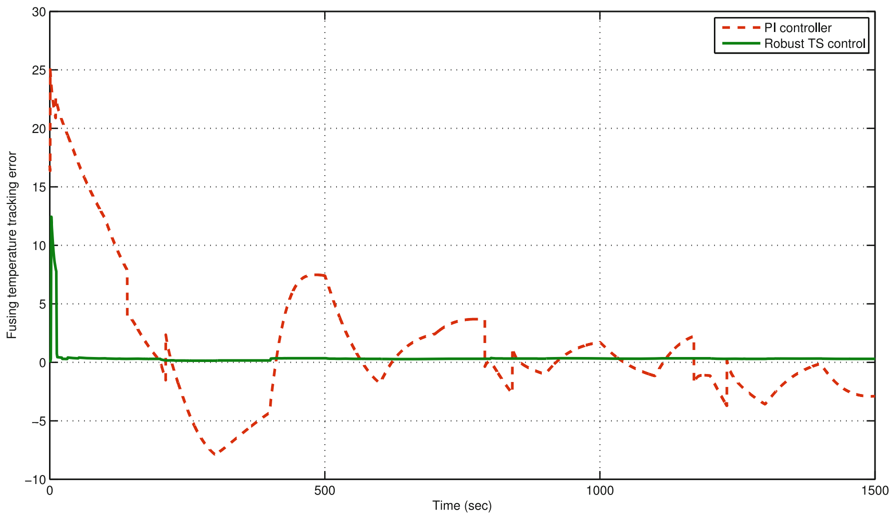


Fig. 4.15 Fusing temperature tracking error

variation in media/print queue (mass of the paper m, initial temperature of the paper T_{init}, dimensions l and b, specific heat capacity c), environment temperature T_{env} and variations in constraints, such as maximum available power.

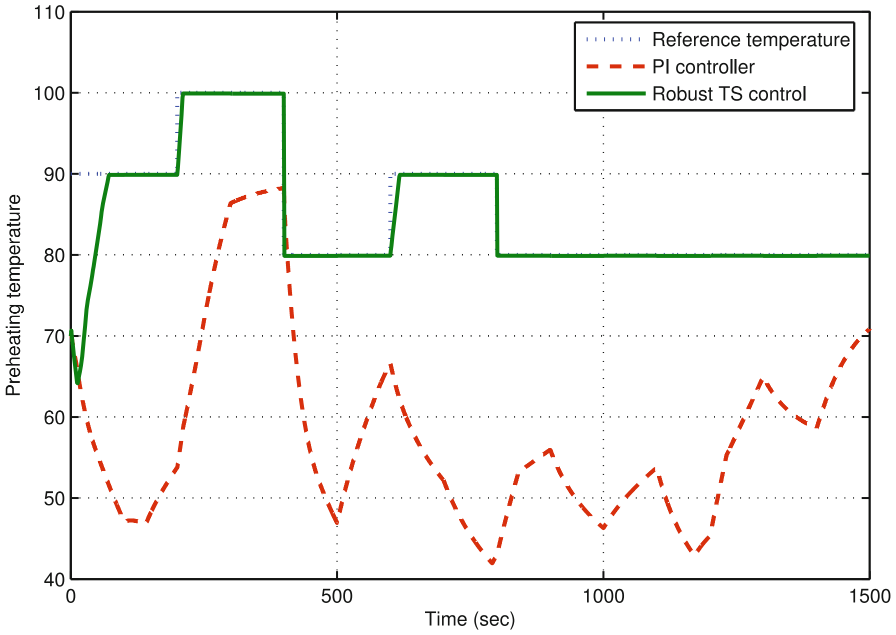


Fig. 4.16 Preheating temperature tracking control comparison

For the warm process, we will illustrate the results obtained for variation in paper mass (since this influences system dynamics the most) and variation in maximum available power (one of the constraints that may not be violated under any circumstances).

4.4.3.1 Centralised Model Predictive Control

The centralised (non-linear) dynamic predictive controller has been designed based on the system characteristics and a known print queue. Since we know in advance what kind of paper will enter the fusing point, we can use this information. The discrete model, to compute the prediction of the output over the prediction horizon, has been obtained from (4.6), (4.7), and (4.12) by discretising with sampling time $T_s = 0.5$ s.

A prediction horizon of $N = 6$ (3 s in advance) is used and for every sample the paper characteristics $\theta(i), i \in I_0^{N-1}$ can be different. In the cost function of the non-linear dynamic controller (4.23) the economic performance is the productivity (4.12) and the output tracking is represented by the fusing temperature T_{fuse} and the preheater temperature T_{pre}. The fusing temperature T_{fuse} has the highest priority $\lambda_1 = 10$ and the preheater temperature T_{pre} has the lowest priority $\lambda_2 = 0.9$, since it is the slowest part of the system and in many cases it can be compensated with

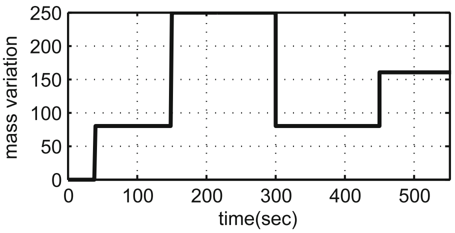


Fig. 4.17 Mass of the paper variation

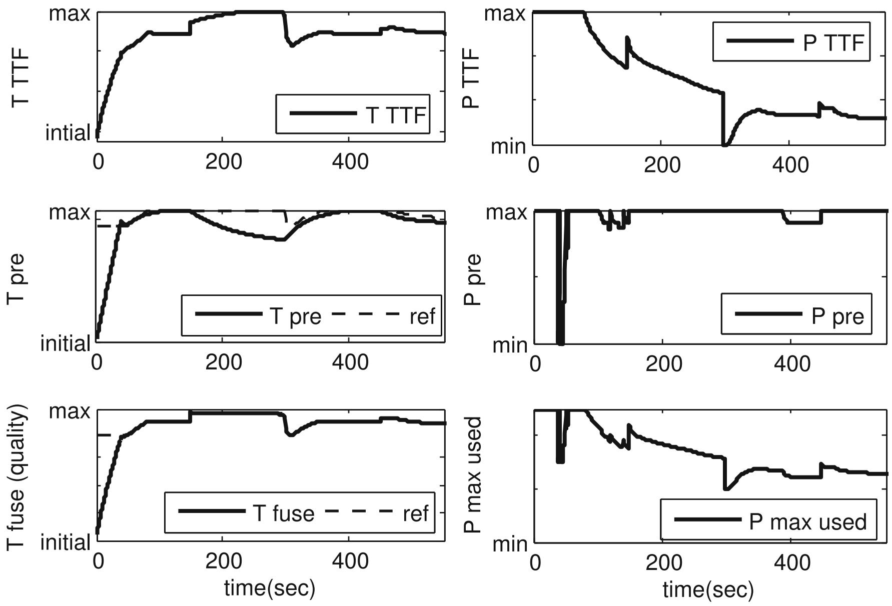


Fig. 4.18 MPC-centralised: Controlled temperatures in the presence of mass variation

the T_{TTF} to obtain the required T_{fuse}. The optimisation is performed on-line and each sample all four inputs $(v, d, P_{\mathrm{pre}}, P_{\mathrm{TTF}})$ are calculated to keep the system's economic performance at a maximum. They are sent directly to the system and at the next sample, based on the output measurements, new inputs will be computed on-line. The unknown disturbances w are some of the paper properties and environment temperature. The outputs y are the measured temperatures of the preheater T_{pre}, the TTF belt T_{TTF}, and the estimated fuse (quality) temperature T_{fuse}.

At cold start, the controller has to determine the initial point, when the printing can start and to heat the printer components as fast as possible to that point. Once $T_{\mathrm{fuse}} \approx T_{\mathrm{ref}}(u_d, \theta)$, determined based on the characteristics of the first sheet and speed, the printing process can start.

Figure 4.17 presents the variation in the mass of the paper between the minimum and maximum allowed $(m_{\min}, m_{\max})[\mathrm{gr}/\mathrm{m}^2]$. For the first seconds, the mass of the paper is zero because the printer is not warm enough, therefore the printing process cannot start yet. Figure 4.18 presents the optimal variation of temperatures inside

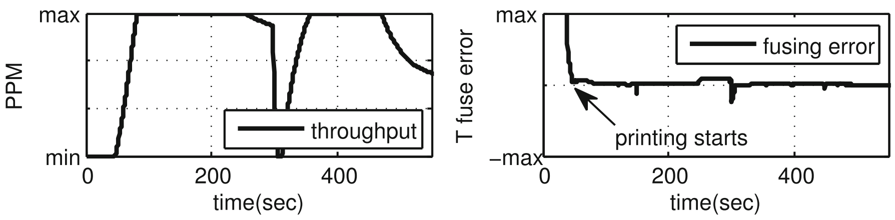


Fig. 4.19 MPC-centralised: Productivity and fuse error for mass variation

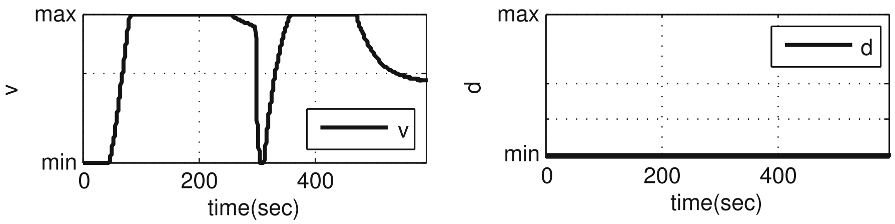


Fig. 4.20 MPC-centralised: Belt speed and distance between sheets for mass variation

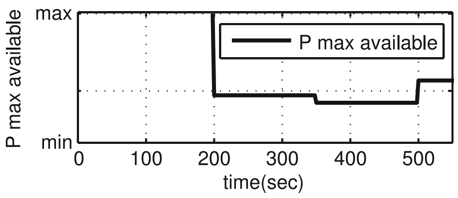


Fig. 4.21 Maximum available power variation

the printer, obtained for the considered case in Fig. 4.17. The plots on the left hand side represent the controlled output of the system to the reference (T_{TTF}, T_{pre}, T_{fuse}). While the plots on the right hand side show the control actions for the TTF and preheater (P_{TTF}, P_{pre}) as well as the maximum used power by the two elements. The references for T_{pre} and T_{fuse} are obtained based on the speed of the engine v and paper properties θ, $y_{\mathrm{ref}}(v, \theta(k))$. The throughput and the error in the fusing point are shown in Fig. 4.19. The throughput has been adjusted on-line according to the available resources. Different combinations of engine speed and distance between sheets can give the same throughput. Since variation in speed of the engine is more energy efficient than adjusting the distance between sheets, the distance between sheets is minimum and speed is varied all the time, see Fig. 4.20. The error in the fuse is very small and remains, after the cold start, always within its bounds.

The maximum available power is an important constraint. The controller must take into account this hard constraint and obtain a feasible, yet optimal, solution. Figure 4.21 presents random steps in the maximum available power. The control scheme knows this maximum, but no prediction is possible. Figures 4.22–4.24 show the robustness and performance of the control scheme with respect to the maximum available power. The fuse quality error in this case is almost zero.

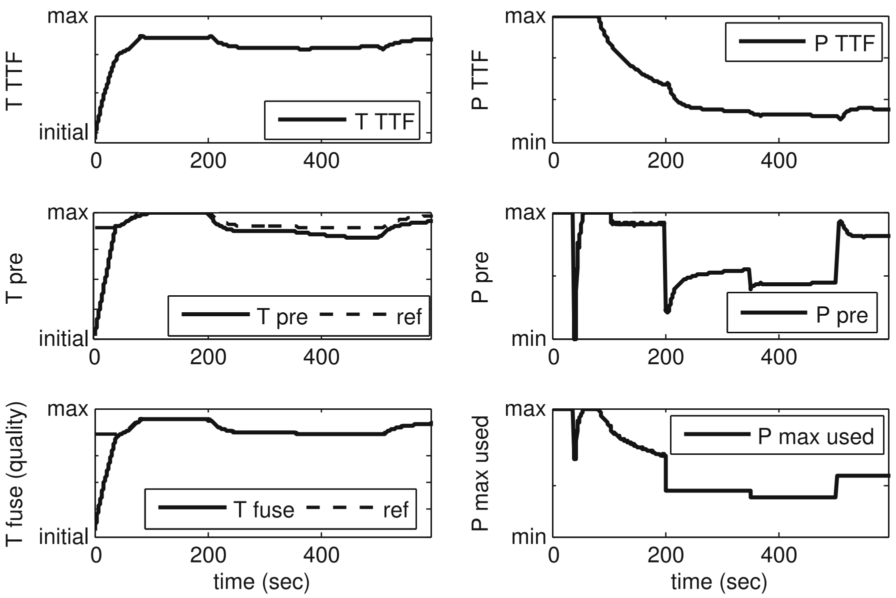


Fig. 4.22 MPC-centralised: Temperatures in presence of $P_{\max}$ variation

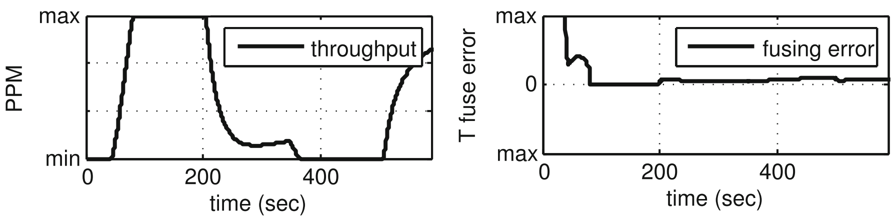


Fig. 4.23 MPC-centralised: Productivity and fuse error in presence of $P_{\max}$ variation

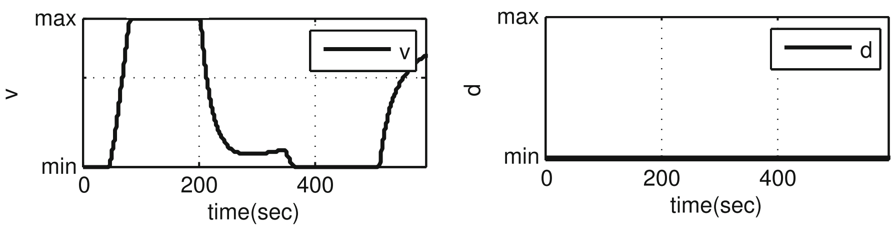


Fig. 4.24 MPC-centralised: Belt speed and distance between sheets with $P_{\max}$ variation

The disadvantage of this control scheme is that the computation time is large for large prediction horizons. Then, the optimal value of the input u cannot be calculated in real-time. For the same considered case studies, the performance of the decomposed MPC will be analysed in the next section.

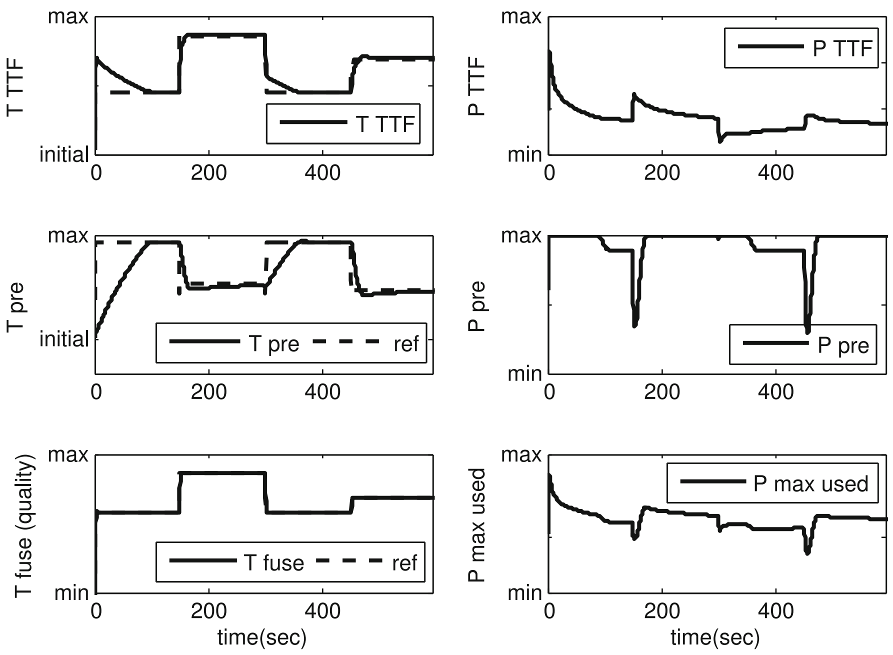


Fig. 4.25 MPC-decomposed: Temperatures in presence of mass variation

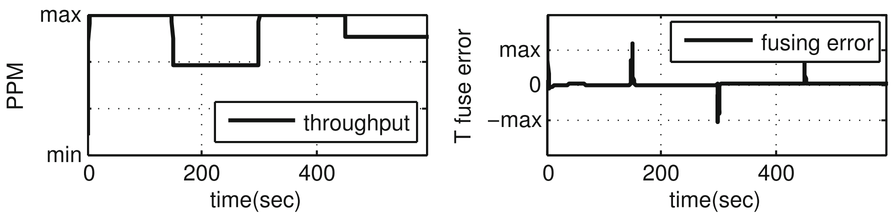


Fig. 4.26 MPC-decomposed: Productivity and fuse error for mass variation

4.4.3.2 Decomposed Model Predictive Control

The results of the decomposition of the plant model are presented next. The reference generator calculates each sample time k the optimal steady-state input $(v, d, P_{\text{pre}}, P_{\text{TTF}})$, based on the non-linear static model. The decoupled inputs, v and d, are sent directly to the system in feedforward. The MPC controller manages to track the setpoints accurately, denoting the robustness of the controller. The system remains stable under all conditions and delivers high performance. The computation time in this case is not a problem (maximum 0.15 seconds per sample) and the linear MPC is robust enough to compensate for modelling uncertainties.

Figures 4.25–4.27 show the results for the variation in the mass of the paper from Fig. 4.17. In this case, the references are calculated by the reference generator.

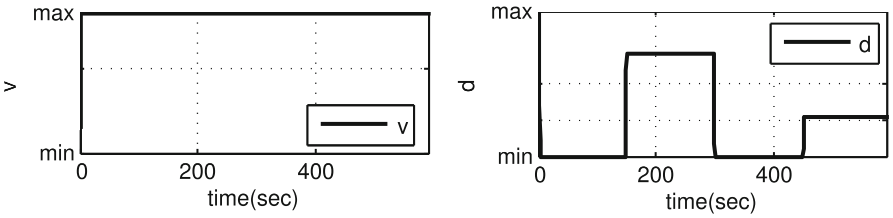


Fig. 4.27 MPC-decomposed: Belt speed and distance between sheets with mass variations

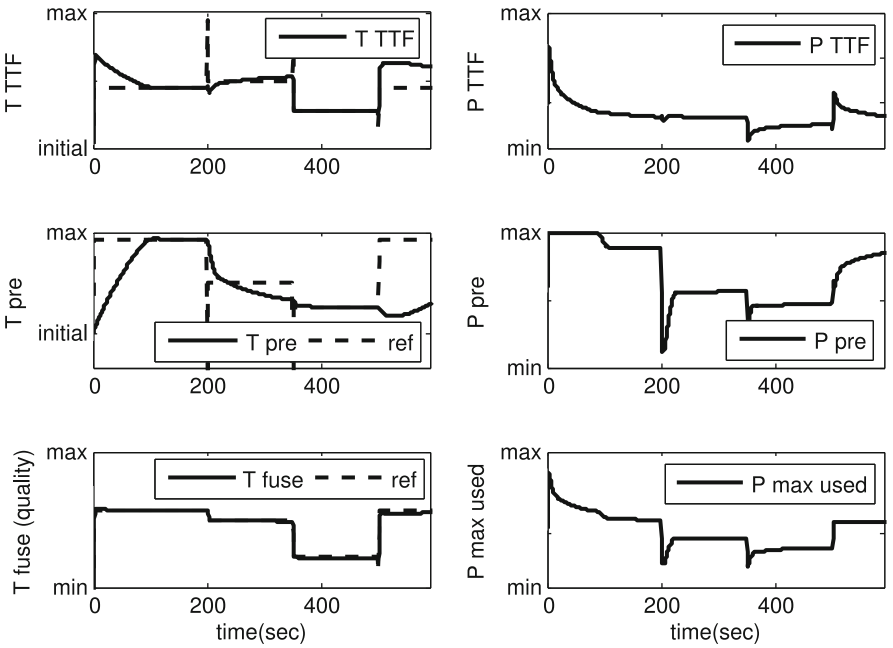


Fig. 4.28 MPC-decomposed: Temperatures in presence of $P_{\max}$ variation

The throughput has been adjusted to fulfil all requirements in a steady-state situation. The error in fuse quality is within specifications most of the time except for some transient situations. As we can see, for the case when the paper characteristics do not change so often, the system can be considered linear and then the decomposed controller will give a good performance.

Figures 4.28–4.30 show the results for the variation in the maximum available power from Fig. 4.21. The throughput will be adjusted to fulfil all requirements in steady state. The error in fuse quality is within specs most of the time.

To conclude this section, the implementation of both MPC strategies shows an improved performance of the printing system. Since the centralised MPC scheme solves a centralised non-linear dynamic optimisation problem, it requires

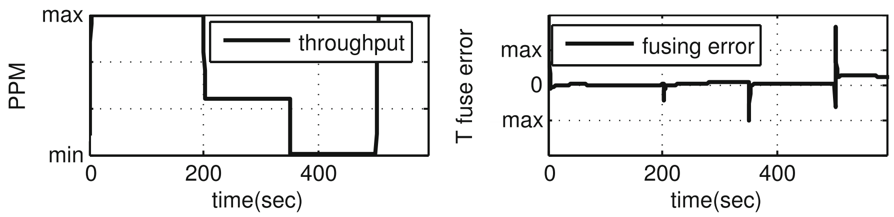


Fig. 4.29 MPC-decomposed: Productivity and fuse error in presence of $P_{\max}$ variation

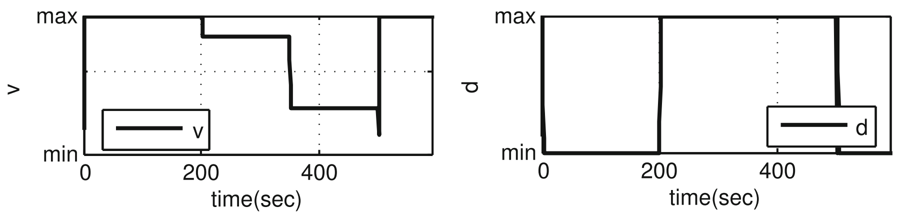


Fig. 4.30 MPC-decomposed: Belt speed and distance between sheets with $P_{\max}$ variation

a large computation time. The computation time can be reduced by decomposing the non-linear dynamic optimisation into a non-linear static optimisation and linear dynamic optimisation. That makes the decomposed MPC scheme more attractive for real-time implementation for the printing system. However, in case of large transitions of media type, the optimality of the solution is sacrificed in the transient performance due to the static optimisation.

4.5 Conclusions

The control design considerations have been discussed for selecting appropriate design methods for adaptive embedded systems, especially for professional toner printers. It has been shown that this class of systems is characterised by a deterministic physics-based model with changing and partly unknown parameters, many constraints, and partly-known disturbances. It has been argued that this class of systems benefits from deterministic control approaches as stochastic approaches cannot sufficiently exploit the existing knowledge. When comparing the deterministic control design methods, as shown in Fig. 4.31, adaptive methods are preferred as they can adapt their control behaviour to the changing operating conditions of a printer. Adaptation allows accurate control. The only requirement is that the adaptation is fast enough to follow the changing operating conditions. To speed up convergence, two new adaptation mechanisms for model reference adaptive control (MRAC) have been described and demonstrated. One mechanism is based on a non-linear gain and the second one is based on multiple gains depending on the operating condition. Especially, when the changing operating condition is

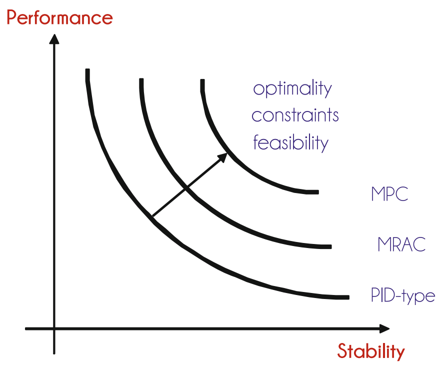


Fig. 4.31 Comparison of different control strategies

known (which is partly true for industrial printers), the multiple gain solution has a clear advantage, as it can adapt faster to changes in the print queue.

Another approach is based on adaptive model predictive control (MPC). This control approach allows the inclusion of all relevant knowledge to maximise the throughput of the printer while still satisfying the strict constraints on print quality and power consumption. Moreover, MPC allows the inclusion of knowledge of the known future disturbance owing to the print jobs in the print queue. A number of examples show the benefits of the new proposed approaches. It can be expected that applying MPC for the industrial electro-photography printer will improve performance considerably compared to the present control approach, based on PI-control and many heuristics. Besides an improved performance, so a higher throughput while satisfying the constraints, the design cycle of the controller will become shorter and leads to more structured design methodology. This means that in one consistent and transparent problem formulation all aspects concerning the operation of a printer can be taken into account. A solution of this large optimisation problem is feasible in real-time and yields the optimal compromise between all conflicting requirements.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. Astrom, K.J., Wittenmark, B.: Adaptive Control. Addison-Wesley, Boston (1995)
2. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
3. Butler, H.: Model Reference Adaptive Control: From Theory to Practice. Prentice Hall International, Upper Saddle River (1992)
4. Cao, S., Rees, N., Feng, G.: Analysis and design of fuzzy control systems using dynamic fuzzy global models. Fuzzy Sets Syst. **75**, 47–62 (1995)

5. Cochior, C., van den Bosch, P.P.J., Waarsing, R., Verriet, J.: Control strategy for systems with input-induced nonlinearities: A printing system case study. In: Proceedings of the 2012 American Control Conference (ACC 2012), Montreal, pp. 1949–1954 (2012)
6. Dessauer, J.H., Clark, H.E.: Xerography and Related Processes. The Focal Press, New York (1965)
7. Ezzeldin, M.: Performance improvement of professional printing systems: From theory to practice. Ph.D. thesis, Eindhoven University of Technology, Eindhoven (2012)
8. Ezzeldin, M., van den Bosch, P.P.J., Waarsing, R.: Improved convergence of MRAC design for printing system. In: Proceedings of the 2009 American Control Conference (ACC 2009), St. Louis, pp. 3232–3237 (2009)
9. Ezzeldin, M., Weiland, S., van den Bosch, P.P.J.: Robust $\mathscr{L}_2$ control for a class of nonlinear systems: A parameter varying Lyapunov function approach. In: Proceedings of the 19th Mediterranean Conference on Control and Automation (MED 2011), Corfu, pp. 213–218 (2011)
10. Ezzeldin, M., Weiland, S., van den Bosch, P.P.J.: Improving the performance of a printing system using model reference adaptive control: An LMI approach. In: Proceedings of the 2012 American Control Conference (ACC 2012), Montreal, pp. 1943–1948 (2012)
11. Ezzeldin, M., Weiland, S., van den Bosch, P.P.J.: Observer-based robust $\mathscr{L}_2$ control for a professional printing system. In: Proceedings of the 2012 American Control Conference (ACC 2012), Montreal, pp. 1955–1960 (2012)
12. Ferramosca, A., Limon, D., Alvarado, I., Almo, T., Camacho, E.F.: MPC for tracking of constrained nonlinear systems. In: Proceedings of the Joint 48th IEEE Conference on Decision and Control (CDC 2009) and the 28th Chinese Control Conference (CCC 2009), Shanghai, pp. 7978–7983 (2009)
13. Fridkin, V.M.: The Physics of The Electrophotographic Process. The Focal Press, New York (1972)
14. Groot, P., Birlutiu, A., Heskes, T.: Learning from multiple annotators with Gaussian processes. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) Artificial Neural Networks and Machine Learning ICANN 2011. Lecture Notes in Computer Science, vol. 6792, pp. 159–164. Springer, Heidelberg (2011)
15. Groot, P., Lucas, P., van den Bosch, P.: Multiple-step time series forecasting with sparse Gaussian processes. In: Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC 2011), Ghent, pp. 105–112 (2011)
16. Heemels, M., Muller, G.: Boderc: Model-Based Design of High-Tech Systems. Embedded Systems Institute, Eindhoven (2007)
17. Krijnen, B.: Heat flow modelling in copiers. Master's thesis, University of Twente, Enschede (2007)
18. Lazar, M., Gielen, R.: On parameterised Lyapunov and control Lyapunov functions for discrete-time systems. In: Proceedings of the 49th IEEE Conference on Decision and Control (CDC 2010), Atlanta, pp. 3264–3270 (2010)
19. Li, D., Xi, Y.: The feedback robust MPC for LPV systems with bounded rates of parameter changes. IEEE Trans. Autom. Control **55**, 503–507 (2010)
20. Lienhard IV, J.H., Lienhard V, J.H.: A Heat Transfer Textbook. The Phlogiston Press, Cambridge (2004)
21. Maciejowski, J.M.: Predictive Control with Constraints. Pearson Education, Upper Saddle River (2002)
22. Magni, L., Scattolini, R.: On the solution of the tracking problem for non-linear systems with MPC. Int. J. Syst. Sci. **36**, 477–484 (2005)
23. MATLAB: http://www.mathworks.com/products/matlab/. Accessed Aug 2012
24. Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M.: Constrained model predictive control: Stability and optimality. Automatica **36**, 789–814 (2000)
25. Nounou, H.N., Passino, K.M.: Stable auto-tuning of the adaptation gain for continuous-time nonlinear systems. In: Proceedings of the 40th IEEE Conference on Decision and Control (CDC 2001), Orlando, pp. 2037–2042 (2001)

26. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modelling and control. IEEE Trans. Syst. Man Cybern. **15**, 116–132 (1985)
27. Tanaka, K., Sugeno, M.: Stability analysis and design of fuzzy control systems. Fuzzy Sets Syst. **45**, 135–136 (1992)
28. Thathachar, M.A.L., Gajendran, F.: Convergence problems in a class of model reference adaptive control systems. In: Proceedings of the 1977 Conference on Decision and Control (CDC 1977), Bangalore, pp. 1036–1041 (1977)
29. Wang, L.: A Course in Fuzzy Systems and Control. Prentice Hall, London (1997)
30. Williams, M.E.: The Physics and Technology of Xerographic Processes. Wiley, New York (1984)
31. Yu, S., Bohm, C., Chen, H., Allgower, F.: Stabilizing model predictive control for LPV systems subject to constraints with parameter-dependent control law. In: Proceedings of the 2009 American Control Conference (ACC 2009), St. Louis, pp. 3118–3123 (2009)

Chapter 5
Reasoning with Uncertainty about System Behaviour: Making Printing Systems Adaptive

Sander Evers, Arjen Hommersom, Peter Lucas, Carmen Cochior, and Paul van den Bosch

Abstract Any conclusion about a system's hidden behaviour based on the observation of findings emanating from this behaviour is inherently uncertain. If one wishes to take action only when these conclusions give rise to effects within guaranteed bounds of uncertainty, this uncertainty needs to be represented explicitly. Bayesian networks offer a probabilistic, model-based method for representing and reasoning with this uncertainty. This chapter seeks answers to the question in what way Bayesian networks can be best developed in an industrial setting. Two different approaches to developing Bayesian networks for industrial applications have therefore been investigated: (1) the approach where Bayesian networks are built from scratch; (2) the approach where existing linear dynamic system theory equations are used as a starting point. A comparison with the traditional method of system identification is also included. The main lesson learned from this research is that Bayesian networks provide a very suitable method for reasoning with uncertainty in engineering problems when detailed engineering knowledge is lacking. When detailed knowledge is available, however, Bayesian networks can still be chosen as an alternative to standard engineering methods. They will offer the advantage that the uncertainty about a problem and its solutions becomes explicit. It will depend on the problem at hand whether their performance will be superior to that of standard engineering methods.

S. Evers • A. Hommersom • P. Lucas (✉)
Department of Model-Based System Development, Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
e-mail: s.evers@cs.ru.nl; arjenh@cs.ru.nl; peterl@cs.ru.nl

C. Cochior • P. van den Bosch
Control Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: c.cochior@tue.nl; p.p.j.v.d.bosch@tue.nl

5.1 Introduction and Motivation

In complex physical systems, often dynamic trade-offs have to be made between their different operating characteristics to ensure that they are functioning properly. This can be viewed as the capability of the system to *adapt* to a changing environment. In printing systems, for example, these characteristics include power supply and consumption, the speed of printing, and the quality of the print-outs. Such trade-offs heavily depend on the system's environment, determined by humidity, temperature, and available power. Failure to adapt adequately to the environment may result in faults or sub-optimal behaviour, leading to low print quality or low productivity.

The problem of adaptability concerns taking actions based on available run-time information, and is also called *decision making*. As defined above it has two main features. First, making decisions is typically required at a low frequency: it is not necessary and not even desirable to change the speed or the energy usage of an engine many times per second. Second, there is a lot of uncertainty involved when making decisions, in particular about the environment, the state of the machine, and also about the dynamics of the system under consideration.

In traditional control engineering methods (cf. Chaps. 3 and 4), such uncertainty is largely ignored. However, when it is taken into account some benefits can be expected. It is instructive to compare the probabilistic approaches that we promote in this chapter against an approach with additive safety margins. In this case a decision variable is not set to the value that would be its theoretical optimum under full knowledge in order to compensate for the uncertainty about other variables that influence the result. For each of these variables, the system designer, therefore, decides on a worst-case deviation, and adds a component to the decision variable's safety margin to counter the effect. Thus, the resulting safety margin is resilient against the combination of worst-case deviations. This conservative approach can be outperformed by any available knowledge about the joint probability distribution of the uncertain variables, as illustrated by Fig. 5.1.

Adaptability requires making system-wide decisions. However, not all components of complex systems can be modelled with the same level of accuracy and precision by means of human expert knowledge. Exploiting machine-learning techniques, by which probability distributions can be estimated from available data, should, therefore, be considered as well. Machine learning can be used either to complement available expert knowledge or as an alternative to expert knowledge when this is difficult to obtain. Building probabilistic models for industrial applications cannot be done effectively without making use of knowledge of engineering methods that are geared to the industrial setting. In this chapter we, therefore, also explore well-known modelling methods from linear dynamic system theory, as commonly used by the engineering community, to facilitate the systematic creation of probabilistic graphical models such as Bayesian networks [14, 17, 33]. In particular, we explore a direction of research where the parameters of a linear dynamic system are assumed to be uncertain. Different options for the representation, inference, and

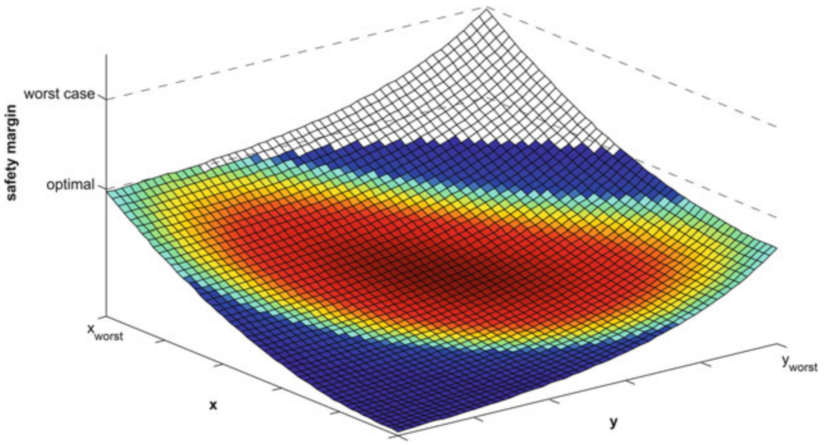


Fig. 5.1 Comparison of a worst-case safety margin and a probability-based safety margin. The values of two uncertain influences x and y are plotted against their combined effect on the safety margin. Assume that it is acceptable to exceed the safety margin in 1% of the cases. Worst-case deviations x_{worst} and y_{worst} have been established based on this criterion. The safety margin labelled "worst case" is based on the simultaneous occurrence of these deviations. The joint probability distribution $\Pr(X,Y)$ is indicated by the colour of the surface (it extends beyond the square shown); although 99% of the probability mass lies in the area where $X < x_{\text{worst}}$, and similarly for Y, it is also true that 99% lies below the mark "optimal". Hence, assuming that a smaller safety margin implies more profit, this is the best choice. In this figure, we have exaggerated the difference between worst case and optimal by choosing a negative correlation between x and y, but the effect is always present to some extent, unless there is a perfect positive correlation

learning of such a system are discussed, and experimental results obtained by this approach are presented. We also compare results obtained with traditional control engineering solutions. The use of Bayesian networks in this context can be seen as decision making about uncertain control parameters, i.e. as *stochastic control*.

Bayesian networks [14, 17, 33] are the main representation and reasoning method used in this chapter to deal with the control of complex systems. Nowadays, Bayesian networks take a central role in dealing with uncertainty in artificial intelligence (AI) and have been successfully deployed in many fields, such as medicine, biology, and finance [28]. The control of physical systems, on the other hand, is largely done using traditional methods from control theory. One of the attractive features of Bayesian networks is that they contain a qualitative part, which can be constructed using expert knowledge, normally yielding an understandable, white-box model. Moreover, the quantitative parameters of a Bayesian network can be estimated from data. Other AI learning techniques, such as neural networks, resist providing insight into why a machine changes its behaviour, as they are black-box models. Furthermore, rules – possibly fuzzy – are difficult to obtain and require extensive testing in order to check whether they handle all the relevant situations.

This chapter summarises our successful effort in using Bayesian-network-based controllers in the industrial design of adaptive printing systems.[1] In our view, as systems get more and more complex, the embedded software will need to be equipped with such AI reasoning capabilities to render the design of adaptive industrial systems feasible.

5.2 Related Work

With respect to decision making, adaptive controllers based on Bayesian networks have not been extensively investigated. The most closely related work is by Deventer [6], who investigated the use of dynamic Bayesian networks for controlling linear and non-linear systems. The premise of this work is that the parameters of a Bayesian network can be estimated from a deterministic physical model of the system. In contrast, we aim at using models that were learned from data. Such data can be obtained from measurements during design time or during run-time of the system.

Several approaches to traditional adaptive control already exist (cf. Chap. 4). First, *model reference adaptive control* uses a reference model that reflects the desired behaviour of the system. On the basis of the observed output and of the reference model, the system is tuned. The second type of adaptive controllers are so-called *self-tuning controllers*; they estimate the correct parameters of the system based on observations and tune the control accordingly. Our approach employs a mixture of the two, where a reference model is given by a Bayesian network and other parts of the system are tuned accordingly. In the last few decades, also techniques from the area of artificial intelligence, such as rule-based systems, fuzzy logic, neural networks, evolutionary algorithms, etc. have been used in order to determine optimal values for control parameters (see e.g. [8]). The main advantages of Bayesian networks over these other uncertainty methods are that they support representing a problem domain in terms of available statistical dependence and independence information, which can be seen as the minimum amount of information needed for model construction, giving rise to graphical models that after some training are easy to understand by humans.

5.3 Background: Bayesian Networks and Linear Dynamic Systems

Bayesian networks and linear dynamic systems are the mathematical notions used in the research described in this chapter. Both are briefly introduced.

[1]This chapter builds on and extends previously published work, in particular [7, 12, 13].

5.3.1 Bayesian Network Models

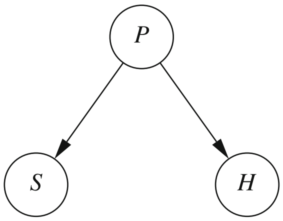
A *Bayesian network* $B = (G, \Pr)$ consists of a directed acyclic graph $G = (V, E)$, where V is a set of nodes and $E \subseteq V \times V$ is a set of directed edges or arcs, and Pr is a joint probability distribution associated with a set of random variables X_V that correspond one-to-one to the nodes V of G, i.e. to each node $v \in V$ corresponds exactly one random variable $X_v \in X_V$ and vice versa [14]. As the joint probability distribution Pr of the Bayesian network is always factored in accordance to the structure of the graph G, it holds that:

$$\Pr(X_V) = \prod_{v \in V} \Pr(X_v \mid X_{\pi(v)}),$$

where $\pi(v)$ is the set of parents of v, i.e. the set of nodes that directly precede v: $\pi(v) = \{u \mid (v, u) \in E\}$. Thus, Pr can be defined as a family of local conditional probability distributions $\Pr(X_v \mid X_{\pi(v)})$, for each node $v \in V$. To abbreviate notation, we will often refer to a variable X rather than the variable and its value $X = x$ if the value does not matter. Instead of the notation $X = x$, we will also simply use x if the value is referred to. Bayesian networks can encode various probability distributions. Most often the variables are either all discrete or continuous. Hybrid Bayesian networks, however, containing both discrete and continuous conditional probability distributions are also possible. A commonly used type of hybrid Bayesian network is the conditional linear Gaussian model [4, 20]. Efficient exact and approximate algorithms have been developed to infer probabilities from such networks [3,22,24]. Also important in the context of embedded systems is that real-time probabilistic inference can be done using anytime algorithms [10].

As a toy example, consider the Bayesian network shown in Fig. 5.2. This model expresses that heat production H and speed of the belt S are dependent of power supply P, where in addition heat production and speed of the belt are conditionally independent given that the power supply if known. This conditional independence statement should be interpreted as telling that when we know with absolute certainty what the power supply is, then knowing as well the heat production tells us nothing *extra* about the speed of the belt and vice versa.

Based on the joint probability distribution Pr of the Bayesian network, assumed to be discrete, and defined according to Fig. 5.2, it is possible to compute any probability of interest by *conditioning* on observations or evidence, such as $\Pr(p \mid h)$, the probability of high power supply assuming that it is certain that the heat production is high. Where $\Pr(P)$ is called the *prior distribution* in this case, $\Pr(P \mid h)$ is called the *posterior distribution* for H taking the value *true* (there is also a distribution, $\Pr(P \mid \neg h)$, for H taking the value *false*). This probability distribution can be computed as follows. Using *Bayes' Theorem*: $\Pr(p \mid h) = \Pr(h \mid p) \cdot \Pr(p) / \Pr(h) = \alpha \cdot 0.8 \cdot 0.95 = 0.98$, where $\alpha = 1/\Pr(h)$ is a normalising constant. Thus, the probability that the power supply is high is very large if we know that the heat production of the heater is high – the *evidence*. We can also



$\Pr(s \mid p) = 0.99$
$\Pr(s \mid \neg p) = 0.2$
$\Pr(h \mid p) = 0.8$
$\Pr(h \mid \neg p) = 0.3$
$\Pr(p) = 0.95$

Fig. 5.2 Simple Bayesian network used to illustrate the basic concepts. The following abbreviations are used: P stands for power (with p "high" and $\neg p$ "low"); S stands for speed of the belt (with s and $\neg s$ the same as for P); H stands for heat production at the heater with the same meaning as the other variables. The *left part* shows the acyclic directed graph and the *right part* the associated discrete conditional probability distributions

compute any joint probability distribution; according to the network structure it holds that $\Pr(p, \neg s, h) = \Pr(\neg s \mid p) \cdot \Pr(h \mid p) \cdot \Pr(p) = 0.01 \cdot 0.8 \cdot 0.95 = 0.0076$, hence the combination of low belt speed, high power availability, and high heat production occurs rarely. Finally, it is possible to compute any probability of any sub-combination of variables or individual variable by *marginalisation*, e.g. $\Pr(s) = \sum_P \Pr(s \mid P) \Pr(P) = 0.99 \cdot 0.95 + 0.2 \cdot 0.05 = 0.9505$.

A Bayesian network can be constructed with the help of domain experts [17]. The graph structure of a Bayesian network facilitates the assessment of probabilities, even to the extent that reliable probabilistic information can be obtained from experts (e.g. [27]). However, building Bayesian networks using expert knowledge can be very tedious and time consuming. Learning a Bayesian network from data is also possible, a task which can be separated into two subtasks: (1) *structure learning*, i.e. identifying the topology of the network, and (2) *parameter learning*, i.e. determining the associated joint probability distribution, Pr, for a given network topology. In this chapter, we employ parameter learning only. This is typically done by computing the maximum likelihood estimates of the parameters, i.e. the conditional probability distributions, associated to the networks structure given data [21], but some other, iterative methods will also come along.

Temporal Bayesian networks are Bayesian networks where the nodes of the graph are indexed with (discrete) time. All nodes with the same time index form a so-called *time slice*. Each time slice consists of a static Bayesian network and the time slices are linked to represent the relationships between states in time. If the structure and parameters of the static Bayesian network are the same at every time slice (with the exception of the first), one speaks of a *dynamic* Bayesian network, as such networks can be unrolled (cf. [5,32]). Temporal Bayesian networks are used in this chapter to model the stochastic dynamics of various processes.

Probabilistic inference is often used for reasoning about hidden states in a dynamic model. Typical applications are *filtering* – i.e. inferring the current hidden state given the observations in the past – and *smoothing* where past states are inferred. For example, the Kalman filter [15] is well known in stochastic control theory (see e.g. [2]) and is a special case of a dynamic Bayesian network, where the model is the linear Gaussian variant of a hidden Markov model, i.e. it describes a Markov process with noise parameters on the input and output variables. Non-linear variants, such as the extended Kalman filter or the unscented Kalman filter (see e.g. [31]) are approximate inference algorithms for non-linear Gaussian models by linearisation of the model. More recently, particle filters [25] have been proposed as an alternative, which relies on sampling to approximate the posterior distribution.

The difference of Bayesian networks with these filtering approaches is that for Bayesian networks there is an underlying domain model which is easy to understand; such is less true for filtering. As Bayesian networks are general formalisms, they can be used for many different purposes. In particular, they have been used for building model-based *diagnostic systems* for various problems, such as for printer troubleshooting [35]. A further advantage compared to black-box models is that the modelled probability distribution can also be exploited for decision making using decision theory. This is particularly important if one wishes to make trade-offs in professional printers such as between productivity and energy consumption.

5.3.2 Linear Dynamic Systems

In its most basic form, a *Linear Dynamic System*, abbreviated LDS, describes a system's behaviour by a differential equation

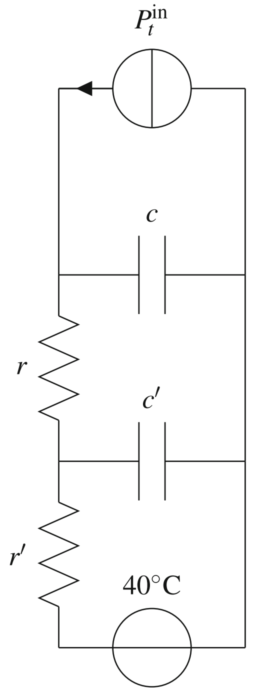
$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

known as a *state-space representation*. Here, vector $\mathbf{x}(t)$ represents the system's state at time t, $\mathbf{u}(t)$ its input at time t, and matrices $\mathbf{A}$ and $\mathbf{B}$ describe how the current *state change* depends linearly on the current state and input. This is a continuous-time representation; in order to calculate with LDS models, time is usually discretised. In this chapter, we therefore use the simple discretised model

$$\mathbf{x}_{t+1} - \mathbf{x}_t = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t$$

in which the size of the discretisation steps conveniently equals the unit of the time domain in order to simplify the exposition (in practice one can use discretisation steps of size Δt and scale the $\mathbf{A}$ and $\mathbf{B}$ matrices with this factor). The equation can then be rewritten to:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{A}_d\mathbf{x}_t + \mathbf{B}_d\mathbf{u}_t \\ \mathbf{A}_d &= \mathbf{A} + \mathbf{I} \\ \mathbf{B}_d &= \mathbf{B}. \end{aligned} \tag{5.1}$$



$$\begin{tikzcd}
 & P_t^{\text{in}} \arrow[dr, "1/c"] & \\
T_t \arrow[rr] \arrow[dr, "1/r"'] & & T_{t+1} \\
 & P_t^{\text{trans}} \arrow[ur, "-1/c"] \arrow[dr, "1/c'"] & \\
T'_t \arrow[ur, "-1/r"] \arrow[rr] \arrow[dr, "1/r'"'] & & T'_{t+1} \\
 & P_t^{\text{out}} \arrow[ur, "-1/c'"] & \\
40^\circ\text{C} \arrow[ur, "-1/r'"] & &
\end{tikzcd}$$

Fig. 5.3 LDS model of a paper heater. *Left*: as an electrical circuit diagram, where voltage represents temperature and current represents heat flow. From *top to bottom*, the diagram consists of a (1) time-variable heat source P_t^{in}, (2) heat mass with capacity c, (3) heat resistance r, (4) heat mass with capacity c', (5) heat resistance r', (6) heat sink with temperature 40 °C modelling the constant flow of paper. *Right*: as a graphical representation of the state-space equations for two discrete time points t and $t+1$. The state of the system is described by the T variables, which represent the temperatures of the two heat masses. Its input is P_t^{in}, the power entering the system. Auxiliary variables represent the power flowing from the first heat mass to the second (P_t^{trans}), and from the second to the heat sink (P_t^{out}). Parameters of the system are the two resistances r and r' and the two capacities c and c' (instantiated with actual values in a real system)

In the engineering process, LDS models of systems are often represented by means of diagrams. We exemplify the role of these models and diagrams in the first case study. The case study originated from Océ. To ensure print quality, paper needs to be heated to a certain temperature, which is accomplished by passing the paper along a metal heater plate, the preheater. It is quite important that the paper reaches the right temperature. If it is too cold, print quality suffers; if it is too hot, energy (and money) is wasted or worse: the printer may malfunction. Therefore, engineers have put a lot of effort in the accurate modelling of the heating process. This results in models such as in Fig. 5.3, in which the heater is modelled as two distinct heat masses: when the heater is powered, the first mass is directly heated thereby indirectly heating the second mass, which transfers the heat to the paper.[2] In the diagram, the heating dynamics are represented as an *electrical circuit*, where temperature plays the part of voltage and heat flow that of current.

Diagrammatic methods for LDSs, as introduced above, have a firmly established place in the engineering process. Important advantages of an electrical diagram and variants, such as schematic mechanical diagrams and bond graphs [16], are:

- They are well suited for *documentation and communication*. A trained engineer can read the system's basic dynamic behaviour from this diagram in a blink of an eye; for a non-expert with a science background it is relatively easy to gain some intuition.
- They have a *formal meaning* as an LDS; it translates into the state-space equations in the form of Eq. (5.1), connecting it to a vast body of theoretical and practical knowledge.
- They *separate qualitative and quantitative aspects* of the model; the former are determined by the diagram structure, the latter by the parameters.
- They are *composable*: other models like this can be developed independently and joined into a larger system.
- They are *supported by software*: drawing and managing modules of electrical circuits and the variants mentioned above can be done by tools like 20-sim [1], which can also perform the translation to the state-space representation. This representation can be used for simulation, e.g. in MATLAB [30].

However, these kinds of diagrams are confined to modelling deterministic behaviour.

5.4 Bayesian Networks in Engineering

In the realm of probabilistic modelling, the Bayesian network formalism shares the attractive properties of LDSs as described above:

- *Documentation and communication*. In scientific literature on machine learning, artificial intelligence, and statistics, Bayesian networks are the de facto pictorial representation of a probabilistic model, and often complement the definition of the probability function. They help experts to quickly understand the structure of this function, and non-experts to see the different ways in which one variable can directly or indirectly influence another.
- *Formal meaning*. Bayesian networks can be mathematically understood in terms of probability distributions.

[2]This is known as a *lumped-element model*; in contrast, the heat distribution could also be modelled as a 3-dimensional temperature field over the plate.

- *Separation of qualitative and quantitative aspects.* When the local conditional probability functions are left out from a Bayesian network, one can still infer formal statements about conditional independence between variables. In an informal (but helpful) way, arrows can often be interpreted as causal influences. The conditional probability functions then quantify the strength of these influences (or specific combinations).
- *Composability*. By their nature, Bayesian networks specify how to compose a global probability distribution from local probability distributions. There also exist formalisms for building large Bayesian networks from components which are Bayesian networks themselves [18, 19].
- *Software support*. Various academic and commercial software packages are available that support model definition, structure and parameter learning, and inference for Bayesian networks.

Thus, a sensible question is whether and how Bayesian networks can be used in engineering for representing problems classes that are probabilistic. In our research, we have tried to answer this question by exploring two different approaches. In the first approach, we have manually constructed a Bayesian network, guided by the general principle of distinguishing different categories of variables, such as control, sensor, state, and goal. In the second approach, we have partially derived the Bayesian network from existing LDS models. These two different ways to build Bayesian networks in engineering are subsequently discussed.

5.4.1 Bayesian-Network Modelling of a Dynamic System

Common assumptions in the modelling of dynamic physical systems are that the system is Markovian and stationary (e.g. [23]), i.e. the system state at time $t+1$ only depends on the system state at time t (*first-order Markov assumption*), and the transition probabilities are the same at each time t (*stationarity assumption*). Stationarity is an assumption too strong for the work discussed in this chapter; however, it is assumed that the network *structure* is the same for every t. In case a particular dependence is absent in a time slice, then such independence will be reflected in the conditional probability distributions rather than in the structure.

Four different types of nodes were distinguished in developing Bayesian networks for stochastic control:

- **Control variables** C act as input to the physical system's control system, such as the car engine's throttle position.
- **Hidden state variables** H determine the *unobservable* state of the system, such as the engine's speed.
- **Sensor information** S provides observations about the (unobservable) state of the machine (here engine), for example by the measurement of the speed of the car.

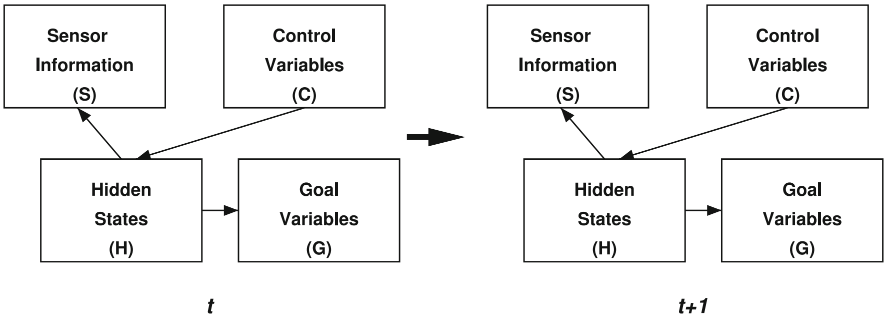


Fig. 5.4 A temporal Bayesian network structure for modelling dynamic systems

- **Target or goal variables** G act as reference values or setpoints of the system. It is the purpose of a Bayesian network to control these variables.

A schematic representation of such a network is shown in Fig. 5.4. Given n time slices, a Bayesian network will have an associated joint probability distribution of the following form:

$$\Pr(S_1, C_1, H_1, G_1, \ldots, S_n, C_n, H_n, G_n).$$

The chosen representation closely fits the concepts of traditional control theory. A typical feedback controller influences the system (H) through a system's input (C); it does so by comparing the sensed data (S) with a reference value (G). A feedforward controller is similar in this view, except that the sensor variables are missing or cannot be observed.

After t time steps, the probability distribution can be updated with the observations $S_1, \ldots, S_t$ and earlier control choices $C_1, \ldots, C_t$ to a probability distribution over the remaining variables:

$$\Pr(H_1, \ldots, H_n, G_1, \ldots, G_n, S_{t+1}, \ldots, S_n, C_{t+1}, \ldots, C_n \mid S_1, \ldots, S_t, C_1, \ldots, C_t).$$

In the following, this conditional probability distribution is abbreviated to

$$\Pr_t(H_1, \ldots, H_n, G_1, \ldots, G_n, S_{t+1}, \ldots, S_n, C_{t+1}, \ldots, C_n). \tag{5.2}$$

A common question in control is to provide an estimation of the goal or target variable, i.e. to compute $\Pr_t(G_k)$ for some k from the conditional probability distribution (5.2). If this can be done reliably, it offers the possibility to exploit the network for control. The controller is able to decide what to do in the future by reasoning about the goal of control in the future $G_f = G_{t+1}, \ldots, G_{t+m}$, where $t + m \leq n$, given a possible choice of control $C_f = C_{t+1}, \ldots, G_{t+p}$, where $t + p \leq n$. Both m and p can be tuned to domain-specific requirements.

Let $U : G_f \to \mathbb{R}$ be a utility function defined for the goal variables G_f. The *expected utility* for controlling the machine by $C_f = c_f$, i.e. $\mathrm{EU}(c_f)$, is then equal to:

$$\mathrm{EU}(c_f) = \sum_{G_f} \Pr_t(G_f | c_f) U(G_f).$$

This approach can also be adapted to continuous variables by integrating over the domain of G_f. A control strategy c_f^* with maximal expected utility yields a maximal value for $\mathrm{EU}(c_f)$:

$$c_f^* = \arg\max_{c_f} \mathrm{EU}(c_f).$$

In Sects. 5.5 and 5.6, we present research in which we have explored the theory summarised above with the aim of making an professional printing system adaptive to its environment.

5.4.2 Bayesian-Network Representation of an LDS

Dynamic Bayesian networks, where the transition probabilities are assumed to be time invariant, offer an appropriate representation for LDSs. For modelling linear dynamic systems as dynamic Bayesian networks, only the following types of nodes are needed:

Deterministic nodes: A node Y with parents $X_1, X_2, \ldots$ is called *deterministic* if its conditional probability distribution is

$$\Pr(y | x_1, x_2, \ldots) = \begin{cases} 1 & \text{if } y = f(x_1, x_2, \ldots) \\ 0 & \text{if } y \neq f(x_1, x_2, \ldots) \end{cases}$$

for a certain function f; in this chapter, these functions are mostly *linear*, i.e.

$$y = \mu + \alpha x_1 + \beta x_2 + \ldots.$$

We use a special notation for these *linear deterministic* nodes shown in Fig. 5.5 (left), where we label the arrows with the linear coefficients, just as in signal flow diagrams [29].

Linear Gaussians: A node Y with parents $X_1, X_2, \ldots$ is known in Bayesian network literature as a *linear Gaussian* if

$$\Pr(Y | x_1, x_2, \ldots) = \mathcal{N}(\mu + \alpha x_1 + \beta x_2 + \ldots, \sigma^2).$$

Networks that consist only of linear Gaussians (with $\sigma > 0$) have theoretical significance: their joint distribution is multi-variate Gaussian, and exact inference

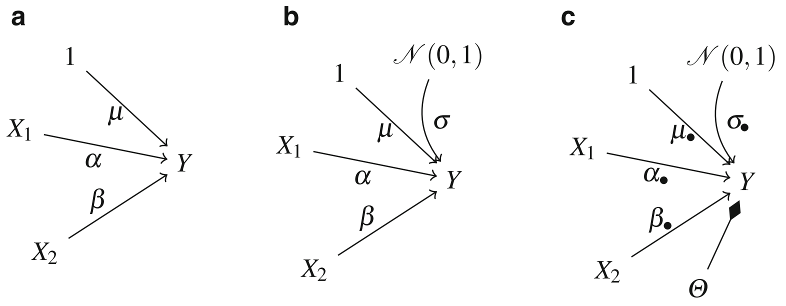


Fig. 5.5 The three basic types of Bayesian network nodes we will use for LDS models. **(a)** Linear deterministic node $\Pr(y|x_1,x_2) = 1$ if $y = \mu + \alpha x_1 + \beta x_2$. **(b)** Linear Gaussian node $\Pr(Y|x_1,x_2) = \mathcal{N}(\mu + \alpha x_1 + \beta x_2, \sigma^2)$. **(c)** Conditional linear Gaussian node $\Pr(Y|x_1,x_2,\theta) = \mathcal{N}(\mu_\theta + \alpha_\theta x_1 + \beta_\theta x_2, \sigma^2_\theta)$ (for discrete Θ). The notation is ours and is introduced in Sect. 5.5.3.2

is easy and efficient (e.g. see [17]). A linear Gaussian without parents $\mathcal{N}(\mu,\sigma^2)$ is simply called Gaussian; the Gaussian $\mathcal{N}(0,1)$ is called a *standard* Gaussian. A linear Gaussian can be written as a linear deterministic node with two extra parents; see Fig. 5.5 (middle).

Conditional linear Gaussians: A node Y with parents $X_1, X_2, \ldots$ and discrete parent Θ is *conditional linear Gaussian* if

$$\Pr(Y|x_1,x_2,\ldots,\theta) = \mathcal{N}(\mu_\theta + \alpha_\theta x_1 + \beta_\theta x_2 + \ldots, \sigma^2_\theta),$$

i.e. it is linear Gaussian for each value θ. If $X_1, X_2, \ldots$ are Gaussian, the marginal distribution over Y is a mixture of Gaussians:

$$\Pr(Y) = \sum_{\theta \in \Theta} \Pr(\theta)\mathcal{N}(\hat{\mu}_\theta, \hat{\sigma}^2_\theta)$$

$$\hat{\mu}_\theta = \mu_\theta + \alpha_\theta \mu_{X_1} + \beta_\theta \mu_{X_2} + \ldots$$

$$\hat{\sigma}^2_\theta = \sigma^2_\theta + \alpha^2_\theta \sigma^2_{X_1} + \beta^2_\theta \sigma^2_{X_2} + \ldots.$$

Again, this also holds for complete networks: if all nodes are (conditional) linear Gaussian, the joint distribution is a mixture of multi-variate Gaussians. However, this number of components in this mixture is exponential in the number of Θ variables, which can make inference hard. A conditional linear Gaussian can also be written as a deterministic node with extra parents. For this we use a special notation shown in Fig. 5.5 (right), to which we will return later.

As these three node types can all be written as deterministic nodes, we will henceforth use the convention that *all non-root nodes in our networks are deterministic*.

5.5 Case 1: Estimation of Media Type

Printing systems contain many challenging control problems. As a first study, we have considered the problem of establishing the media mass during a run, which can be exploited during the control of many different parts of the printer. For example, if the media mass, here interpreted as paper type, is known, then we can (1) avoid bad print quality, (2) avoid engine pollution in several parts of the printer, and (3) help service technicians at service call diagnosis, e.g. to avoid malfunction for specific media under specific circumstances. Given the limitations on development costs, component cost price, and available space in printers, it is deemed infeasible to design a printer that measures paper properties directly, nor is it desirable to ask the user to supply this information. We have therefore investigated whether the available sensor data can be used to estimate these properties. This sensor data mainly consists of temperatures and voltages that are required during the regular control of the printer and is available without any additional hardware costs.

5.5.1 Different Approaches and Their Evaluation

The described problem has been addressed in different ways. First, we have used a Bayesian network, designed with the help of an engineering expert, to tackle the problem. Subsequently, a methodology where an existing LDS was exploited was developed and evaluated. Finally, as estimation of the media type can be interpreted as what is called a *classification problem* in the machine-learning literature, for which many different methods exist, it was decided to be worthwhile to compare some carefully selected methods. It is well known in the machine-learning field that very simple models, that in principle offer a weak representation of the problem domain, are often very good at the classification task. This is known as the *bias-variance trade-off* : even though a model can have a high bias, by which it will be unable to fit to the data really well, it can perform well because of low variance, since it does not overfit to the data (for details consult [11]). Furthermore, in some cases, the problem that needs to be solved is simply easy, so that more complex models have little to contribute.

There are many different methods available to evaluate classification performance. *Accuracy* is simply the percentage of cases that have been classified correctly. One of the more insightful methods is the *receiver operating characteristics* (ROC) curve, where the true-positive ratio (the percentage of correctly classified true cases) is plotted against the false-positive ratio (the percentage of incorrect, i.e. as true classified, false cases) under different preset thresholds on the numeric value produced by a predictive model that, thus, acts as classifier [9]. By computing the *area under the ROC curve*, often abbreviated to AUC, one obtains a single number that allows one to compare two different classifiers. However, the shape of the ROC

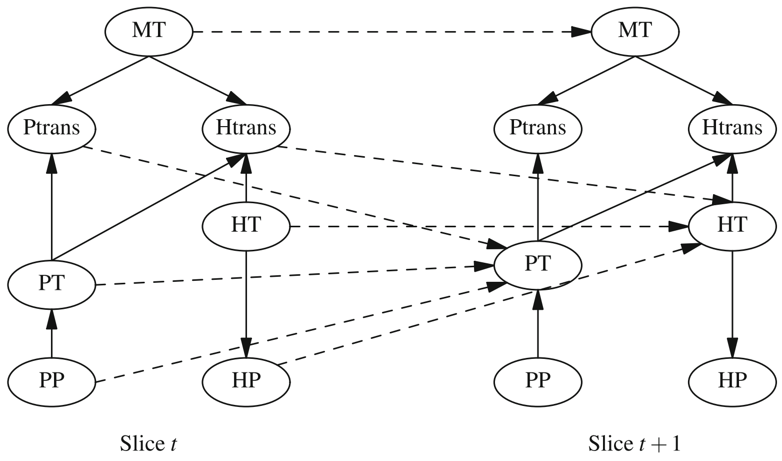


Fig. 5.6 Dynamic Bayesian network that was designed with the help of a domain expert. Used abbreviations are *MT* MediaType, *PP* Power Preheater, *PT* Temperature Preheater, *Ptrans* Power transferred from preheater, *HP* Power Fuse, *HT* Temperature Fuse, *Htrans* Power transferred from fuse

curve is also relevant as it conveys extra information. We compare some of these methods, including a system identification method from control theory, with each other in Sect. 5.5.4.

5.5.2 *Using a Manually Designed Bayesian Network*

The data that were available consisted of logging data of runs from stand-by mode with a warm machine. In order to vary different conditions, the duration of stand-by prior to the run was deliberately varied. This ensures a realistic variation of temperatures inside the engine. Moreover, the paper type was varied, namely the data contain runs with $70\,\text{g/m}^2$ ($n = 35$), $100\,\text{g/m}^2$ ($n = 10$), $120\,\text{g/m}^2$ ($n = 24$), $140\,\text{g/m}^2$ ($n = 10$), and $200\,\text{g/m}^2$ ($n = 29$) paper.

With the help of the domain experts we designed a Bayesian network structure, shown in Fig. 5.6, with seven nodes at each time slice. Logging data of a professional printing system were obtained at a frequency of 2 Hz for 15 s, which would yield a total of 210 nodes in the temporal Bayesian network if represented explicitly. All variables were modelled as Gaussian random variables. To convert the Bayesian network into a classifier, the estimations of the model were mapped to a number of classes, corresponding to the distinguished media types.

The plot shown in Fig. 5.7 indicates that it takes some time before the Bayesian network is able to reliably distinguish between the media types based on the sensor information available. After about 6 s the classification reaches a performance that is seen as sufficiently high for many applications. However, for high-speed printing

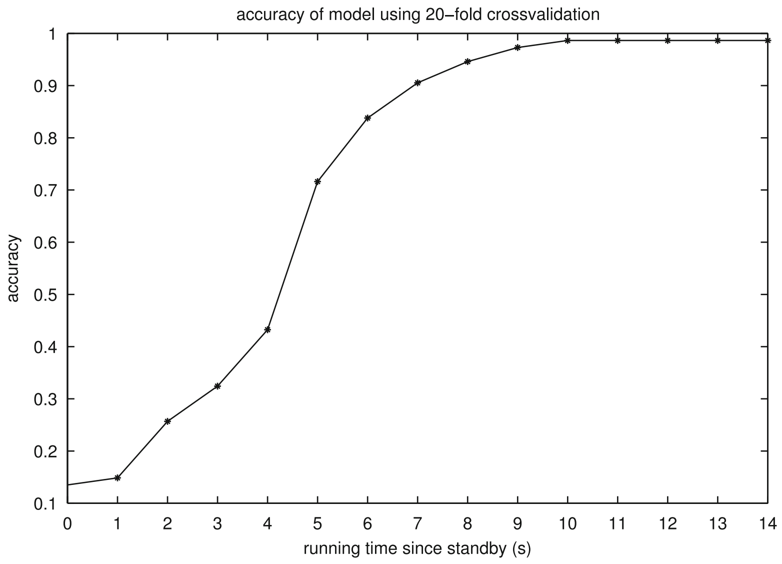


Fig. 5.7 This plot shows the classification accuracy of the Bayesian network plotted as a function of time after the start of a print job

systems a higher reliability may be required. As the plot shows, on the whole there is a gradual, and robust increase in performance with a form that looks like a sigmoid learning curve. However, note that the only thing that changes in time are the data: the nature of the data is such that in time it becomes easier for the Bayesian network to distinguish between various media types. Further evidence of the robustness of the approach is obtained by computing the confidence intervals of the mass estimates. As shown in Fig. 5.8, the confidence intervals become smaller in time, and conclusions about media type are therefore also more reliable. Hence, it is fair to conclude that the model was able to derive useful information about media type by using sensor information, here about temperature and power usage, that is not immediately related to media type.

In case there is reasonable confidence in the computed estimation, a decision can be made to adapt the system's behaviour and the method will be of practical value.

5.5.3 Using a Dynamic Bayesian Network Derived from an LDS

In this section, the methodology that supports the development of a Bayesian network based on an available LDS, and that was sketched in Sect. 5.4.2, is illustrated by the means of the case study.

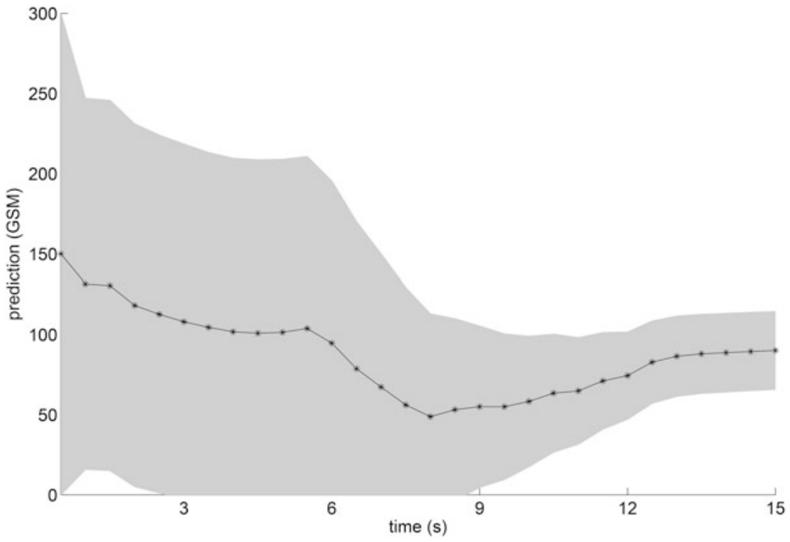


Fig. 5.8 The estimation of paper mass plotted in time. The *solid line* denotes the mean media mass estimation and the *grey area* visualises three standard deviations from the mean

5.5.3.1 An LDS for the Paper Heater System

The paper heater model in Fig. 5.3 translates to the following discrete-time state-space equations in the form of Eq. (5.1):

$$\begin{aligned}\begin{bmatrix} T_{t+1} \\ T'_{t+1} \end{bmatrix} &= \mathbf{A}_d \begin{bmatrix} T_t \\ T'_t \end{bmatrix} + \mathbf{B}_d \begin{bmatrix} P^{\text{in}}_t \\ 40\,^\circ\text{C} \end{bmatrix} \\ \mathbf{A}_d &= \begin{bmatrix} 1 - 1/rc & 1/rc \\ 1/rc' & 1 - 1/rc' - 1/r'c' \end{bmatrix} \\ \mathbf{B}_d &= \begin{bmatrix} 1/c & 0 \\ 0 & 1/r'c' \end{bmatrix}. \end{aligned} \tag{5.3}$$

Recall that the variables r and r' stand for heat resistances, c and c' for heat capacities, and P^{in}_t is the heat power source. The state of the system consists of the temperatures T_t and T'_t of the two heat masses. In fact, translating the electrical diagram by tools such as 20-sim [1] first leads to a more elaborate form in which auxiliary power variables are present. This system of linear equations is shown in a graphical form (a signal flow diagram) at the right side of Fig. 5.3:

- Each *node* represents the left-hand side of an equation, consisting of one variable.
- The incoming *arcs* represent the right-hand side: a linear combination of the parent variables, with coefficients as specified on the arcs. Note: we follow the convention that empty arcs carry a coefficient of 1.

For example, the figure shows that

$$P_t^{\text{trans}} = \frac{1}{r} T_t + \frac{-1}{r} T_t' = \frac{T_t - T_t'}{r},$$

$$T_{t+1} = \frac{1}{c} P_t^{\text{in}} + T_t + \frac{-1}{c} P_t^{\text{trans}} = T_t + \frac{P_t^{\text{in}} - P_t^{\text{trans}}}{c}.$$

The state-space Eq. (5.3) are obtained from this system by substituting the P^{trans} and P^{out} variables for their right-hand sides.

Represented in this graphical form, the system of linear equations *already completely defines a Bayesian network*, albeit one without any stochastic influences: *each node is deterministic*. Note that, because we adopted the signal flow notation with linear coefficients on the arrows, the graph can be interpreted both as a conventional system of linear equations and as a Bayesian network.

The next step, then, is to add uncertainty to the LDS. First, as is often done (e.g. in the Kalman filter), we augment the state variables with additive zero-mean Gaussian noise:

$$\begin{bmatrix} T_{t+1} \\ T_{t+1}' \end{bmatrix} = \mathbf{A}_d \begin{bmatrix} T_t \\ T_t' \end{bmatrix} + \mathbf{B}_d \begin{bmatrix} P_t^{\text{in}} \\ 40\,^\circ\text{C} \end{bmatrix} + \begin{bmatrix} W_t \\ W_t' \end{bmatrix}$$

$$W_t \sim \mathcal{N}(0, \sigma^2)$$

$$W_t' \sim \mathcal{N}(0, \sigma'^2) \tag{5.4}$$

The noise is represented by two independent variables W_t and W_t'. A graphical representation of this system is shown in Fig. 5.9a; we have only replaced W_t and W_t' by two anonymous standard Gaussian variables $\mathcal{N}(0,1)$ whose value is multiplied by σ and σ'. As a result, the T_t variables now have the linear Gaussian form from Fig. 5.5 (middle).

In fact, this makes the whole system Gaussian: although the P_t^{trans} and P^{out} nodes are not linear Gaussian, we have already seen that they can be marginalised out, making the T_{t+1}, T_{t+1}' nodes directly depend on T_t, T_t'. As for the P_t^{in} node: as it is always used with concrete evidence p_t^{in}, it never represents a probability distribution, and can be reformulated to take the place of μ in the T_{t+1} distribution. Thus, Fig. 5.9a is a dynamic Bayesian network with a Gaussian joint distribution. This means that we can use a variant of the standard Kalman filter [15, 17] to do exact inference on this network.

5.5.3.2 LDS Models with Uncertain Parameters

Above, we have shown how to represent an LDS with additive zero-mean Gaussian noise as a Bayesian network; while it may be instructive, thus far it is only a

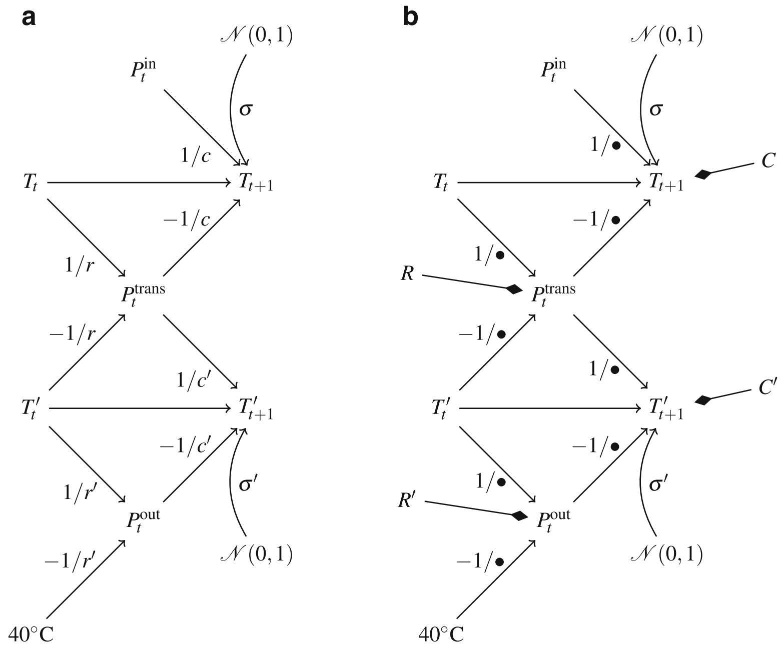


Fig. 5.9 (**a**) LDS of Eq. (5.4), containing additive zero-mean Gaussian noise on the state variables; (**b**) LDS of Eq. (5.4) augmented with uncertain parameters. These are modelled by *conditional linear deterministic* nodes

convenient reformulation of known theory. However, to model the relation with the paper type variable, we need uncertainty in the *parameters*, i.e. the coefficients of the linear relation. Our solution is to transform these coefficients into probabilistic variables. We accomplish this by introducing a new node type:

Conditional linear deterministic node: A linear deterministic node extended with a special parent, which is distinguished from the rest because its arc ends in a diamond (and carries no coefficient); we call this parent the *conditioning parent*. The coefficients on the other arcs *can depend on the value of the conditioning parent*. This dependence is shown by putting a bullet in the place where this value is to be filled in.

We have already given an example of such a node: the conditional linear Gaussian node in Fig. 5.5. Just like the linear Gaussian node is an instance of a linear deterministic node, viz. having specific parents 1 and $\mathcal{N}(0,1)$, a conditional linear Gaussian node is a specific instance of conditional linear deterministic node.

By using conditional linear deterministic nodes, we can extend our already noisy paper heater model with uncertain parameters: we replace parameter r by variable R – which becomes the conditioning parent of the node that depended on r – and do the same for the other parameters. The result is shown in Fig. 5.9b.

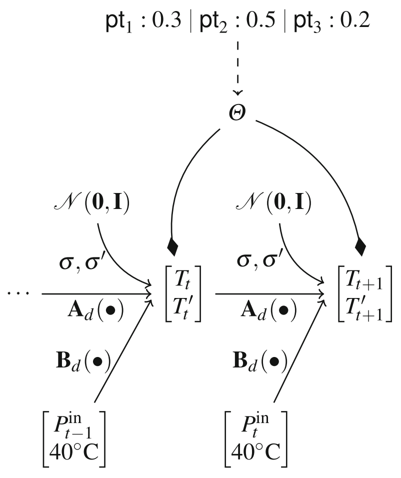

$\mathsf{pt}_1 : 0.3 \mid \mathsf{pt}_2 : 0.5 \mid \mathsf{pt}_3 : 0.2$
Θ
$\mathcal{N}(\mathbf{0},\mathbf{I})$
σ,σ'
$\mathbf{A}_d(\bullet)$
$\mathbf{B}_d(\bullet)$
$\begin{bmatrix} T_t \\ T'_t \end{bmatrix}$
$\begin{bmatrix} T_{t+1} \\ T'_{t+1} \end{bmatrix}$
$\begin{bmatrix} P^{\text{in}}_{t-1} \\ 40°\text{C} \end{bmatrix}$
$\begin{bmatrix} P^{\text{in}}_{t} \\ 40°\text{C} \end{bmatrix}$

Fig. 5.10 The model from Fig. 5.9b, summarised using vector variables (where Θ represents R, C, R', C') and augmented with a paper type variable with a discrete distribution. The relation between time slices $t-1$ and t is also shown. There are several options for the relation between paper type and Θ (the *dashed arc* here is not a linear influence)

We have subsequently proceeded by connecting a discrete paper-type variable to the model, with a distribution of pt_1, pt_2, and pt_3 equal to the probabilities 0.3, 0.5, and 0.2, respectively. Like we mentioned, the paper type determines the R' parameter, but for generalisation's sake we will assume that it influences all the parameters. The resulting model, in Fig. 5.10, also shows that the notation for (conditional) linear deterministic nodes extends very naturally to vector-valued variables: coefficients become matrices. These are the matrices from Eq. (5.3), written as functions $\mathbf{A}_d(\theta)$ and $\mathbf{B}_d(\theta)$ of $\theta = (r, c, r', c')$. Thus, we have made a *graphical summary* of the model which is linked very clearly to the state-space equations. Although this hides some of the model's internal structure, it is useful for keeping an overview.

Regarding the probability distribution of the Θ variable, we consider two possibilities:

Discrete Θ: The paper types pt_1, pt_2, pt_3 deterministically set a value θ_1, θ_2, θ_3 (respectively) for Θ. In fact, this turns T_t and T_{t+1} into conditional linear Gaussians (conditioned by the paper type), so the joint distribution is a mixture of three Gaussians.

Continuous Θ: The paper types pt_1, pt_2, pt_3 determine the parameters $(\mu_{\theta 1}, \sigma_{\theta 1})$, $(\mu_{\theta 2}, \sigma_{\theta 2})$, $(\mu_{\theta 3}, \sigma_{\theta 3})$ for a Gaussian-distributed Θ. This model is no longer linear.

These options have an influence on inference and learning in the model, which we discuss in the next sections.

It is interesting to compare learning for continuous and discrete Θ. In order to do this, we have simulated the system in Fig. 5.3 for 150 time slices, with a sine wave as P^{in} input and random Gaussian disturbance. We provide the EM algorithms discussed above with the P^{in} and the generated T'_t data. Typical results of a 60-iterations run are shown in Fig. 5.11.

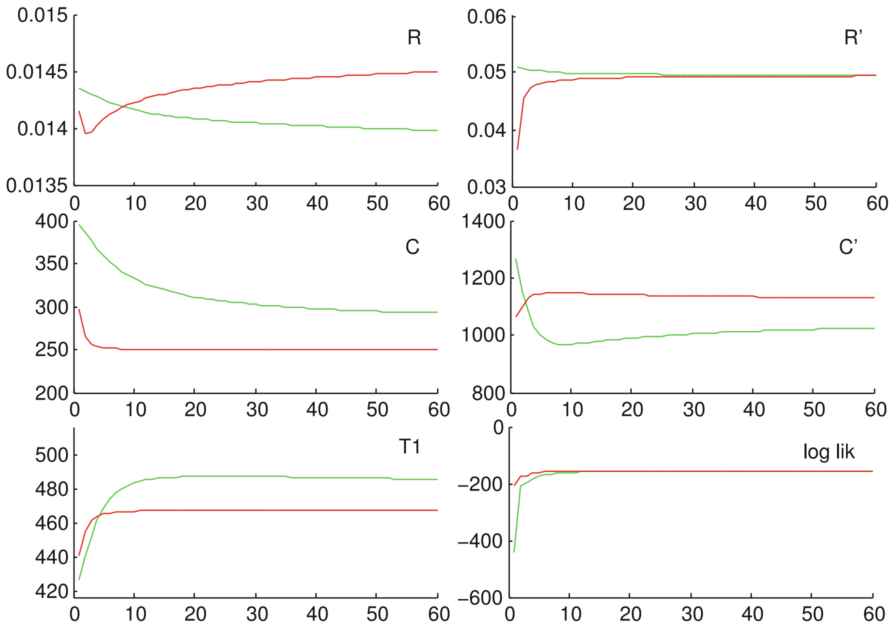


Fig. 5.11 EM learning of the Θ parameters: comparison of discrete parameter space (parameters are single values; shown in *green*) and continuous parameter space (parameters are Gaussians; μ values shown in *red*). The horizontal axis represents 60 EM iterations. Also shown are the learned distribution over T_1 (Gaussian, μ value) and the log likelihood

The most interesting fact to observe is that the two approaches converge to different values (but the same log likelihood). This probably means that the system is not *identifiable*: several choices for Θ lead to the same likelihood of the observed behaviour T_t'. To test this hypothesis, we also generated synthetic T_t, T_t' data (without disturbance) for systems with the learned parameters. The results are plotted in Fig. 5.12. These results indeed show that both methods arrive at the same approximation (green, red) of the original (blue) T_t' data; however, the different values for the parameters lead to a different behaviour of T_t.

5.5.4 A Comparison between Three Models Used as Classifiers

A *classifier* is a piece of software that can be regarded as a function $f : O \rightarrow C$ that given particular observations O is able to label these observations from a finite set of elements C, called *classes*. When labelling data generated by a dynamic process, classification has a temporal dimension: $f : O \times \mathscr{T} \rightarrow C$, where $\mathscr{T}$ stands for the time axis. In the context of the work described in this chapter,classifiers are studied

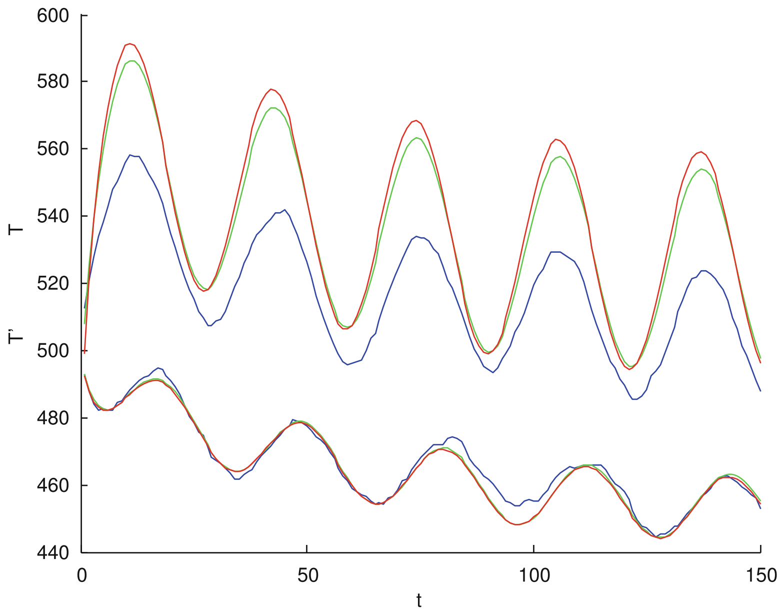


Fig. 5.12 *Blue:* Synthetic data used for learning (T_t and T'_t are shown, but only the latter is given to the learning algorithms). As input P_t^{in}, we used a sine wave. We disturbed the system with additive zero-mean Gaussian noise. *Green*, *red*: Response of an undisturbed deterministic system using the learned parameters (with discrete and continuous parameter space, respectively) to the same P_t^{in} sine wave

that act as a kind of *virtual sensors* to determine paper type based on information about available power and paper temperature, i.e. the function looks as:

$$P(0), T'(0), \ldots, P(t), T'(t) \rightarrow \{80, 120, 200\}$$

for each time instant t. The three possible output values correspond to the mass (gsm = grams per square metre) of the three different paper types we want to distinguish.

In contrast to the Bayesian networks described earlier, the classifiers in this section deal with situations where the paper type can change without warning; it is the task of the classifier to detect at what moment this happens, and to which of the predefined classes the new paper type belongs. We go through classifiers from very simple to more sophisticated. We used time-series data for three runs of a printer that allowed us to compare the performance of the different methods.

5.5.4.1 Steady-State Equation

After some time of heating the same type of paper, the system described by the model in the left part of Fig. 5.3 will reach *steady state*. This means that the

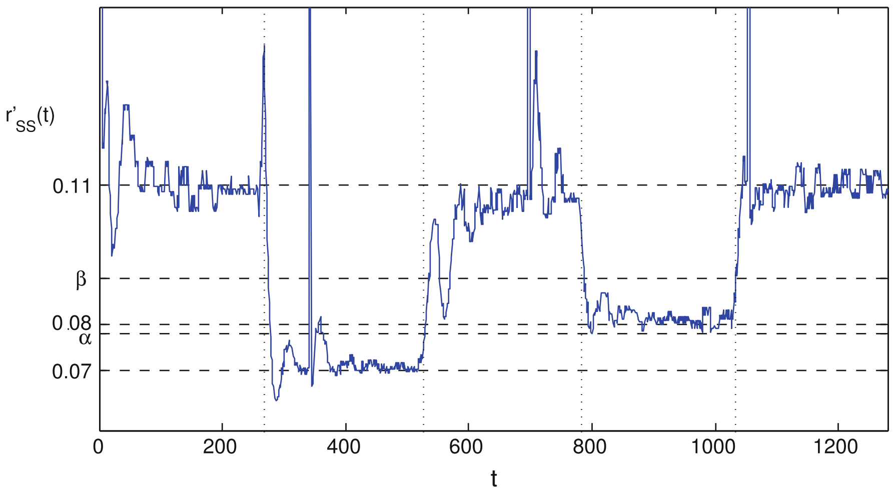


Fig. 5.13 Steady-state thermal resistance r'_{SS} for the first paper run. The three different paper masses $\{80, 120, 200\}$ lead to clearly discernible values of 0.11, 0.08, and 0.07, respectively. We have chosen thresholds $\alpha = 0.09$ and $\beta = 0.078$ that optimise accuracy on this run

temperatures $T(t)$ and $T'(t)$ do not change anymore (and $T'(t)$ will be at the controller setpoint). Thus, the net power flow into/out of both thermal masses is 0 and all power $P(t)$ goes into the paper. This enables us to determine r':

$$r'_{\mathrm{SS}}(t) = \frac{T'(t) - 40\,^\circ\mathrm{C}}{P(t)}.$$

On inspection of the first paper run data, we saw that this value converges to 0.11, 0.08, and 0.07 for 80, 120, and 200 gsm paper, respectively. We will use these values in subsequent sensors which try to determine, at each moment, the current value of r'.

We can also use the steady-state equation itself as a virtual sensor, i.e. ignore all the dynamics and determine r' at each time as if the system were in steady state. By choosing two thresholds $0.07 < \alpha < 0.08 < \beta < 0.11$, we can define a virtual sensor that selects one of the paper types based on this $r'_{\mathrm{SS}}(t)$:

$$\mathrm{gsm}_{\mathrm{SS}}(t) = \begin{cases} 80 & \text{if} \qquad\qquad\qquad\qquad \beta < r'_{\mathrm{SS}}(t) \\ 120 & \text{if} \qquad\quad \alpha < \; r'_{\mathrm{SS}}(t) < \; \beta \\ 200 & \text{if} \; r'_{\mathrm{SS}}(t) < \; \alpha \end{cases}$$

Note that this virtual sensor only makes use of the latest $T(t)$ and $P(t)$ values, ignoring all of the history.

In Fig. 5.13, the $r'_{\mathrm{SS}}(t)$ values for paper run #1 are plotted, and thresholds α and β have been chosen to optimise the virtual sensor's accuracy on this run, making use of the knowledge the true paper type (in other words, the first run is used as training data for the α and β parameters of this sensor). We will use the same α and β when we test this sensor on the other runs.

5.5.4.2 Recursive System Identification

Recursive system identification [26] takes a window of w samples $P(t-w+1)$, $T'(t-w+1), \ldots, P(t), T'(t)$ and looks for the model that best fits these samples, in the following sense:

$$\begin{aligned}\hat{\mathbf{x}}(t+1) &= \mathbf{A}(r')\hat{\mathbf{x}}(t) + \mathbf{B}(r')\mathbf{u}(t) + \mathbf{K}e(t)\\ T'(t) &= \begin{bmatrix}0 & 1\end{bmatrix}\hat{\mathbf{x}}(t) + e(t),\end{aligned}$$

where the **A** and **B** matrices are the discretised equivalent

$$\mathbf{A}(r') = \mathbf{I} + s\begin{bmatrix}-1/rc & 1/rc\\ 1/rc' & -1/rc' - 1/r'c'\end{bmatrix} \qquad \mathbf{B}(r') = s\begin{bmatrix}1/c & 0\\ 0 & 1/r'c'\end{bmatrix}$$

of the model above (with s the duration of a discrete time unit). In these matrices, the r, c, and c' parameters are fixed to those values determined from the step response. The r' parameter and gain matrix **K** are free; the virtual sensor thus determines, for the current window, the value r' that minimises the squared error in the equations above (“prediction error minimisation”):

$$r'_{\mathrm{RSI}}(t) = \arg\min_{r'} \sum_{\tau=t-w+1}^{t} e^2(\tau).$$

Again, this is turned into a decision on paper type by thresholding:

$$\mathrm{gsm}_{\mathrm{RSI}}(t) = \begin{cases} 80 & \text{if } \beta < r'_{\mathrm{RSI}}(t)\\ 120 & \text{if } \alpha < r'_{\mathrm{RSI}}(t) < \beta\\ 200 & \text{if } r'_{\mathrm{RSI}}(t) < \alpha\end{cases}$$

We use the same α and β as for the steady-state equation.

5.5.4.3 Switching LDS

In a probabilistic model, we do not simply minimise the error, but capture it by a probability distribution. *Learning* a model then often means choosing the parameters for this distribution that maximise the probability, or its likelihood, of the observed data. Our model is very similar to the discrete model used in the previous paragraph: we use the same $\mathbf{A}$ and $\mathbf{B}$ matrices, with r, c, and c' fixed, but instead of estimating r', we assume that it is determined by a discrete stochastic variable PT_t (paper type), which can change at each instant t and take a value from $\{80, 120, 200\}$.

$$\Pr(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathsf{pt}_t) \sim \mathcal{N}\left(\mathbf{A}(\mathsf{pt}_t)\mathbf{x}_t + \mathbf{B}(\mathsf{pt}_t)\mathbf{u}_t, \begin{bmatrix} 2 & 0 \\ 0 & 0.3 \end{bmatrix}\right)$$

$$\Pr(\mathsf{pt}_{t+1} \mid \mathsf{pt}_t) = \begin{cases} 0.98 & \text{if } \mathsf{pt}_{t+1} = \mathsf{pt}_t \\ 0.02 & \text{if } \mathsf{pt}_{t+1} \neq \mathsf{pt}_t \end{cases}$$

$$\mathbf{A}(\mathsf{pt}_t) = \mathbf{I} + s\begin{bmatrix} -1/rc & 1/rc \\ 1/rc' & -1/rc' - 1/r'c' \end{bmatrix} \qquad \mathbf{B}(\mathsf{pt}_t) = s\begin{bmatrix} 1/c & 0 \\ 0 & 1/r'c' \end{bmatrix}$$

$$\text{where } r' = \begin{cases} 0.11 & \text{if } \mathsf{pt}_t = 80 \\ 0.08 & \text{if } \mathsf{pt}_t = 120 \\ 0.07 & \text{if } \mathsf{pt}_t = 200 \end{cases}$$

Note that the three values for r' are taken from the steady-state equation data in Fig. 5.13 again; the other values (2 and $0.3\,^\circ\mathrm{C}^2$ for the state update variances, 0.02 for the probability of paper change) have been manually determined as acceptable values. For the virtual sensor, we have used probabilistic *inference* to determine the probability distribution over pt_t at each time t, given the observations until time t. The virtual sensor then outputs the paper type with the highest probability:

$$\mathrm{gsm}_{\mathrm{SW}}(t) = \arg\max_{\mathsf{pt}_t} \Pr(\mathsf{pt}_t \mid [\mathbf{x}_\tau]_2 {=} T'(\tau), \mathbf{u}_\tau {=} (P(\tau), 40\,^\circ\mathrm{C}) \text{ for } \tau = 0..t).$$

5.5.4.4 Virtual Sensor Performance

In Figs. 5.14 and 5.15, the output of the different virtual sensors on test runs #2 and #3 is shown. For the switching LDS model, the probabilities for the three different outputs are shown in shades of grey. For each sensor, the accuracy is given.

The $\mathrm{gsm}_{\mathrm{SS}}$ and $\mathrm{gsm}_{\mathrm{RSI}}$ virtual sensors depend on the thresholds determined from the first run. These thresholds can be sometimes chosen poorly: for example, notice in Fig. 5.13 that the α threshold lies very close to the 120 g steady-state mean value. If α would have been chosen closer to the 200 g value, the virtual sensor would have performed worse on the training set, but would be more robust. For example, if the steady-state value for 120 g would drop a little due to unmodelled influences (such as the input paper temperature dropping), the paper would still be classified correctly. This is a classic bias-variance problem; the accuracy of the classifiers is sensitive to variance in the training set.

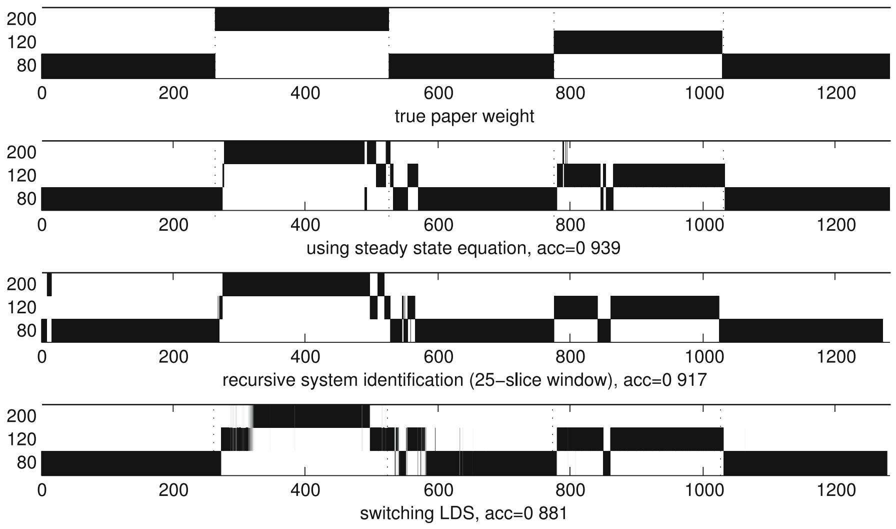


Fig. 5.14 Virtual sensor performance on run #2

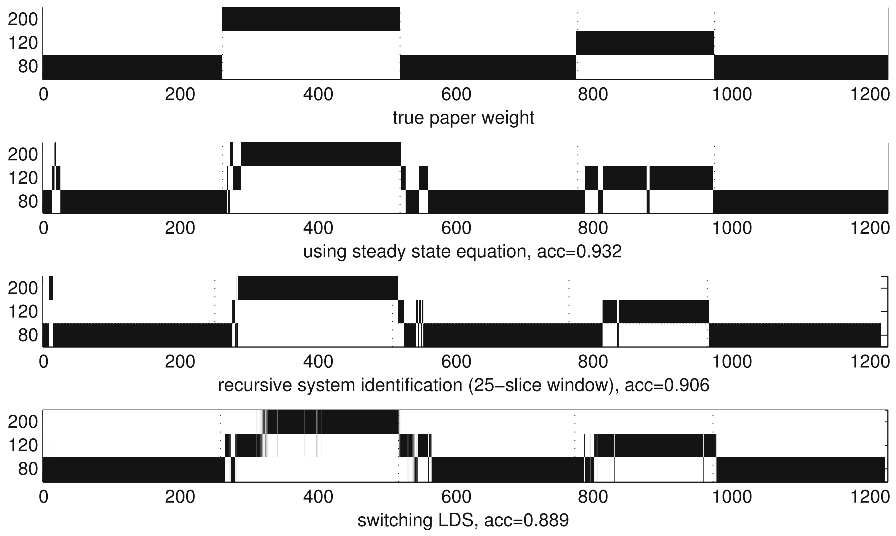


Fig. 5.15 Virtual sensor performance on run #3

A measure that takes into account all possible thresholds is the AUC (area under ROC curve; see Sect. 5.5.1). It is less sensitive to this variance. Also, it does not depend on the true paper type ratios. The average was computed for two ROC curves: 80 versus (120/200) g and 200 versus (80/120) g. The AUC values for the different sensors are:

Method	Run #2	Run #3
SS	0.9564	0.9733
RSI	0.9514	0.9636
SW	0.9669	0.9645

The performance of the different approaches are very close to each other, and one may therefore conclude that each of these methods would be suitable for the classification task. The fact that even the simple steady-state equation performs well indicates that the classification problem is very simple, and a simple rule-based discrimination would already suffice. If there would have been a dynamic deviation from the steady state due to some hidden process, then this would no longer be the case, and the other methods would perform better. However, there were no data available to test this hypothesis.

5.6 Case 2: Control of Engine Speed

The productivity of printers is limited by the amount of power available, in particular in countries or regions with weak mains. If there is insufficient power available, then temperature setpoints cannot be reached, which causes bad print quality. To overcome this problem, it is either possible to decide to always print at lower speeds or to adapt to the available power dynamically. In this section, we explore the latter option by a dynamic speed adjustment using a Bayesian network.

5.6.1 Different Approaches and Their Evaluation

The block diagram in Fig. 5.16 offers an overview of this approach. In this schema, "sensors" are put on low-level controllers and signal the high-level controller with requests. The network then reasons about appropriate setpoints of the low-level controller. In this problem setting, the high-level controller decides on a specific engine velocity based on requested power by a lower-level controller.

For this problem, we discuss the dynamic Bayesian-network model of a specific part of the printer in more detail. The structure of the model at each time slice is shown in Fig. 5.17. The requested power available is an observable variable that depends on low-level controllers that aim at maintaining the right setpoint for reaching a good print quality. The error variable models the deviation of the actual temperature from the ideal temperature, which can be established in a laboratory situation, but not during run-time. If this exceeds a certain threshold, then the print quality will be below the norm that has been determined by the printer manufacturer.

Both velocity and available power influence the power that is or can be requested by the low-level controllers. Furthermore, combining both the available

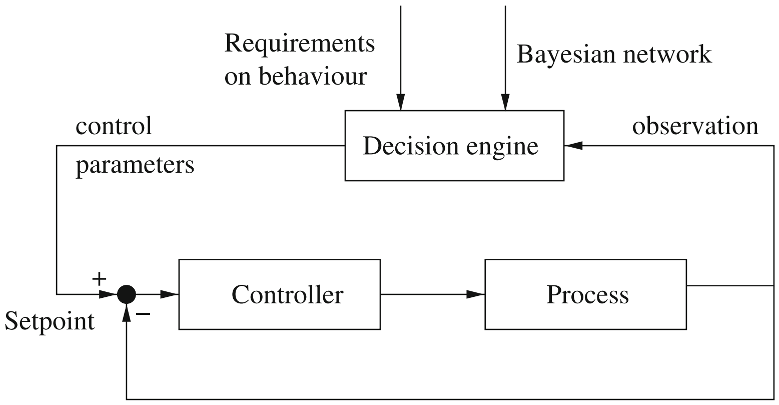
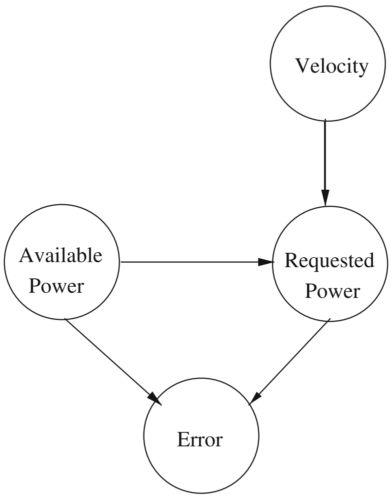


Fig. 5.16 Architecture of an adaptive controller using a Bayesian network

Fig. 5.17 Structure of the Bayesian network of each time slice



and requested power yields a good predictor of the error according to the domain experts. To model the dynamics, we have used two time slices with the interconnections between the available power – which models that the power supply on different time slices is not independent – and requested power, which models the state of the machine that influences the requested power.

We again considered to model all the variables as Gaussian distributed random variables. This seemed reasonable, as most variables were Gaussian distributed, however with the exception of the available power (see Fig. 5.18). Fitting a Gaussian distribution to such a distribution will typically lead to insufficient accuracy. To improve this, this variable was modelled as a mixture of two Gaussian distributed variables, one with mean $\mu_{\text{Power}}^{\text{low}}$ and one with mean $\mu_{\text{Power}}^{\text{high}}$ with a small variance. Such a distribution can be modelled using a hybrid network as follows. The network

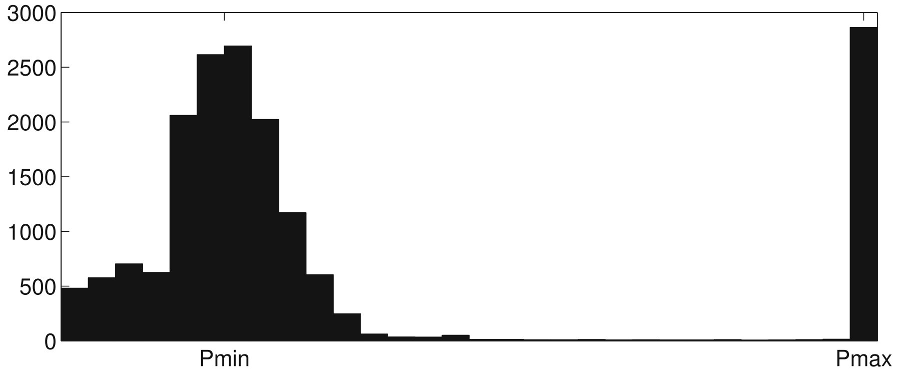


Fig. 5.18 Distribution of the requested power showing skewness and a considerable gap between two high frequency peaks; thus, approximating by a single Gaussian distribution is not a good idea

is augmented with an additional (binary) parent node S with values "high" and "low" for the requested power variable. For both states of this node, a normal distribution is associated to this variable. The marginal distribution of the requested power is obtained by:

$$\Pr(\mathrm{P}_{\mathrm{req}}) = \sum_{S} \Pr(\mathrm{P}_{\mathrm{req}} \mid S) \Pr(S).$$

5.6.2 Error Estimation

The main reasoning task of the network is to estimate the error, i.e. the deviation from the ideal temperature given a certain velocity and observations. This problem can be considered as a classification task of the print quality is *bad* or *good*. The advantage is that this provides a means to compare different models and see how well they perform at distinguishing between these two possibilities. Again ROC and AUC analysis was done (see Sect. 5.5.1).

We have compared three models: (1) a discrete model, (2) a fully continuous model, and (3) a hybrid model for modelling the distribution of the requested power with two Gaussian distributed random variables. The performance is shown in Fig. 5.19. As expected, the fully continuous model performs worst, whereas the hybrid and discrete models show a similar trend. The advantage of the discrete version is that the probability distribution can easily be inspected and it has no underlying assumptions about the distribution, which makes it easier to use in practice. The hybrid version however allows for more efficient computation as we need a large number of discrete values to describe the conditional distributions. For this reason, we have used the hybrid version in the following.

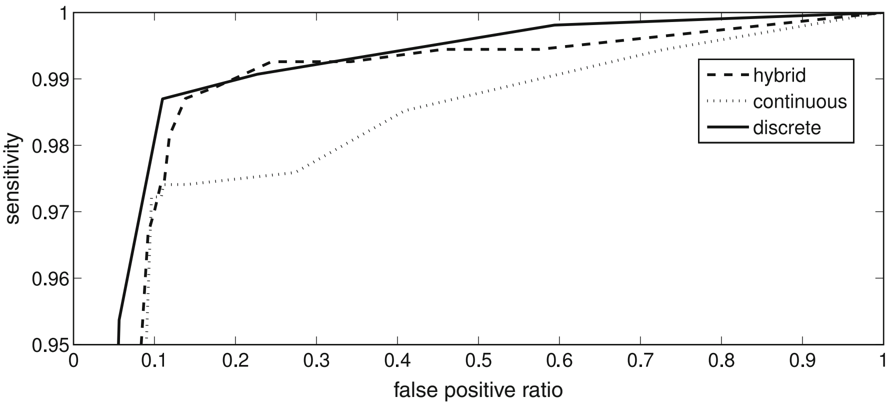


Fig. 5.19 ROC curves of the three Bayesian networks. The hybrid and discrete versions show the best classification performance. Note that the best performance is obtained by a classifier with ROC curve with largest AUC value. However, the shape of the curve also matters. The discrete model obtains good performance at the left part of the curve, i.e. a high true-positive ratio at low false-positive ratio

5.6.3 Decision Making for Control

As the error information is not observable during run-time, the marginal probability distribution of the error in the next time slice is computed using the information about the power available and requested. This error is a Gaussian random variable with mean μ and standard deviation σ. The maximum error that we allow in this domain is denoted by $E_{\max}$ and we define a random variable for print quality Q_k, which is true if $\mu + k\sigma < E_{\max}$, where k is a constant. Different values of k correspond to different points on the ROC curve as depicted in Fig. 5.19. For a Gaussian random variable, more than 99.73% of the real values of the error will be within three standard deviations of the mean, so for example $k = 3$ would imply that $\Pr(\text{Error}_{t+1} < E_{\max}) > 99.87\%$. The goal variables of our control are the print quality, modelled by Q_k (*true* or *false*), and the velocity V. A utility function $U : \{Q_k, V\} \to \mathbb{R}$ defined by

$$U(q,v) = \begin{cases} -1 \text{ if } q = \textit{false} \\ v \quad \text{otherwise} \end{cases}$$

was used to capture the trade-off; optimal control was determined by the maximal expected utility criterion as introduced in Sect. 5.4.1. This implies that the expected utility of a velocity v, $\text{EU}(v)$, equals -1 or v depending on the risk of having bad print quality. In reality, we have chosen the highest velocity v such that $Q_k = \textit{true}$.

To evaluate the model and method, we have compared the productivity of the resulting network with a *rule-based method*, implemented in terms of repetitive and conditional statements of a programming language for use in the actual control

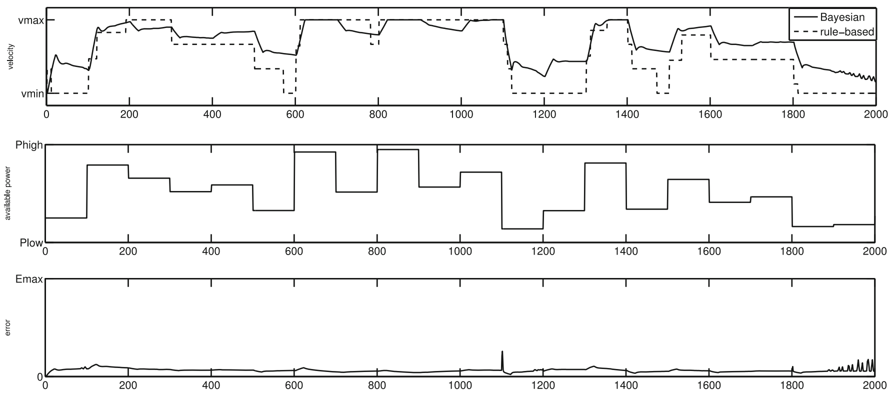


Fig. 5.20 In the *centre* figure, the available power is plotted, which is fluctuating. at the *top*, we compare the velocity of the engine which is controlled by a rule-based system and by a Bayesian network. *Below*, we present the error that the controller based on the Bayesian network yields, which is within the required limits

engine, that incorporated some heuristics for choosing the right velocity. The productivity is defined here simply as $\int_0^\tau v(t)\mathrm{d}t$, where τ is the simulation time. In order to smooth the signal that the network produces, we employ a FIR (Finite Impulse Response) filter in which we employ a moving average of 10 decisions. The resulting behaviour was simulated and is presented in Fig. 5.20 (with $k = 3$). Compared to the rule-based approach, we improve roughly 9% in productivity while keeping the error within an acceptable range. While it is true that the rules can be improved and optimised, the point is that the "logic" underlying the controller does not have to be designed if a Bayesian network is used. What is required is the network structure of a graphical model, data, and a probabilistic criterion that can be inferred.

5.7 Conclusions and Outlook

The first, central, scientific question which initiated our research described in this chapter was whether or not the ability to explicitly handle uncertainty using probabilistic representation and reasoning methods will make it easier to draw conclusions about the behaviour of a physical system. This question has been studied by applying different, but related, methods to two different case studies regarding a professional printing system. The second question addressed in this chapter was to what extent knowledge engineering methods for Bayesian-network models need to be built on existing methods from control engineering when the aim is to develop industrial applications. This question has been explored by using modelling methods that were either only very loosely based on the engineering models that were available, or that were almost completely based on these.

The research in which we developed Bayesian networks based on expert knowledge only – thus these were only loosely based on engineering models – was successful in the sense that the developed models performed very well. For the approach built on existing engineering methods, first a new systematic framework had to be developed. The idea pursued was that linear system theory and associated diagrams and notations can be used as a first step in the knowledge engineering process. Our experience was that this does indeed have the advantage that it facilitates the development of Bayesian network models. One of the reasons for this is that engineers have already dealt with the unavoidable complexity issues in representing realistic models. Subsequently, we have shown how linear dynamic system models can be augmented with uncertain parameters, transforming them into dynamic Bayesian networks. However, a disadvantage of the latter, more restrictive, approach is that it is also more difficult to capture behaviour that is not modelled in the original LDSs. Media recognition was the case where these alternative approaches have been further scrutinised. In a comparison between different methods, ranging from simple rule-based to more sophisticated system identification and Bayesian-network methods, the obvious has again become clear, namely that very simple methods may perform well if the task is easy enough. This only confirms what is well known in the machine-learning community [11]. Nevertheless, it may sometimes be worthwhile to construct models that are more powerful than needed for the task at hand as this may make it easier to reuse them for other tasks. This is particularly true for the model-based approach supported by Bayesian networks.

Finally, it has been shown as well in our research that Bayesian networks can be used very successfully as a framework for adaptive control in physical systems. Here, finding the right trade-off between energy consumption and speed of printing was the case studied.

In embedded software research, there is an increasing trend to apply and verify new software methods in an industrial setting, which is sometimes called the *industry-as-laboratory* paradigm [34]. This means that concrete cases are studied in their industrial context to promote the applicability and scalability of solution strategies under the relevant practical constraints. Much of current AI research, on the other hand, is done within a theoretical setting using standard benchmark problems and data sets for validation. It poses a number of challenges if one wishes to apply AI techniques, such as Bayesian networks, to industrial practice. First, there is little modelling support. Bayesian networks are expressive formalisms and little guidance is given to the construction of networks that can be employed in such an industrial setting. Moreover, there is not a lot of dedicated theory that supports applying Bayesian networks to these areas. For example, while there is a lot of research in the area of stochastic control, it is unclear how these results carry over to Bayesian networks. Similarly, techniques developed in the setting of Bayesian networks do not carry over automatically to the problem of control. Thus, there is more than sufficient space for further research in the area where Bayesian networks meet industrial problems, and so far the results have been promising.

We have shown in this chapter that Bayesian networks can act as a good basis for designing an intelligent, adaptive printing system, sometimes with surprisingly little work. This suggests that Bayesian networks offer interesting promises for engineering applications, in particular in areas such as control and state detection.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. 20-sim tooling. http://www.20sim.com. Accessed Aug 2012
2. Åström, K.J.: Introduction to Stochastic Control Theory. Academic Press, New York (1970)
3. Casella, G., Robert, C.: Monte Carlo Statistical Methods. Springer, New York (1999)
4. Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J.: Probabilistic Networks and Expert Systems. Springer, New York (1999)
5. Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Comput. Intel. **5**, 142–150 (1989)
6. Deventer, R.: Modeling and control of static and dynamic systems with Bayesian networks. Ph.D. thesis, Technische Fakultät der Universität Erlangen-Nürnberg, Erlangen (2004)
7. Evers, S., Lucas, P.J.F.: Constructing Bayesian networks for linear dynamic systems. In: Proceedings of the 8th UAI Bayesian Modeling Applications Workshop (BMAW-11), Barcelona, pp. 26–33 (2011)
8. Farrell, J.A., Polycarpou, M.M.: Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches. Wiley, Hoboken (2006)
9. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. **27**, 861–874 (2006)
10. Guo, H., Hsu, W.: A survey of algorithms for real-time Bayesian network inference. In: Proceedings of the AAAI/KDD/UAI02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems, Edmonton (2002)
11. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2001)
12. Hommersom, A., Lucas, P.J.F.: Using Bayesian networks in an industrial setting: Making printing systems adaptive. In: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010), Lisbon, pp. 401–406 (2010)
13. Hommersom, A., Lucas, P.J.F., Waarsing, R., Koopman, P.: Applying Bayesian networks for intelligent adaptable printing systems. In: M. Conti, S. Orcioni, N. Martínez Madrid, R.E.D. Seepold (eds.) Solutions on Embedded Systems. Lecture Notes in Electrical Engineering, vol. 81, pp. 201–213. Springer, Heidelberg (2011)
14. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, 2nd edn. Springer, Berlin (2007)
15. Kalman, R.E.: A new approach to linear filtering and prediction problems. J. Basic Eng. **82**, 35–45 (1960)
16. Karnopp, D.C., Margolis, D.L., Rosenberg, R.C.: System Dynamics: Modeling and Simulation of Mechatronic Systems, 4th edn. Wiley, Hoboken (2006)
17. Koller, D., Friedman, N.: Probabilistic graphical models: Principles and techniques. MIT, Cambridge (2009)
18. Koller, D., Pfeffer, A.: Object-oriented bayesian networks. In: Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97), Providence, pp. 302–313 (1997)

19. Laskey, K.B.: MEBN: A language for first-order Bayesian knowledge bases. Artif. Intell. **172**, 140–178 (2008)
20. Lauritzen, S.L.: Propagation of probabilities, means and variances in mixed graphical association models. J. Am. Stat. Assoc. **87**, 1098–1108 (1992)
21. Lauritzen, S.L.: The EM algorithm for graphical association models with missing data. Comput. Stat. Anal. **19**, 191–201 (1995)
22. Lauritzen, S., Spiegelhalter, D.: Local computations with probabilities on graphical structures and their application to expert systems. J. R. Statistical Soc. **50**, 157–224 (1988)
23. Lerner, U., Moses, B., Scott, M., McIlraith, S., Koller, S.: Monitoring a complex physical system using a hybrid dynamic Bayes net. In: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI 2002), Edmonton, pp. 301–310 (2002)
24. Lerner, U., Parr, R.: Inference in hybrid networks: theoretical limits and practical algorithms. In: Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI 2001), Seattle, pp. 310–318 (2001)
25. Liu, J.S., Chen, R.: Sequential Monte Carlo methods for dynamic systems. J. Am. Stat. Assoc. **93**, 1032–1044 (1998)
26. Ljung, L.: System Identification: Theory for the User, 2nd edn. Prentice Hall, Upper Saddle River (1999)
27. Lucas, P.J.F., Boot, H., Taal, B.G.: Computer-based decision-support in the management of primary gastric non-Hodgkin lymphoma. Methods Inf. Med. **37**, 206–219 (1998)
28. Lucas, P., Gamez, J.A., Salmerón, A. (eds.): Advances in Probabilistic Graphical Models. Studies in Fuzziness and Soft Computing, vol. 213. Springer, Heidelberg (2007)
29. Mason, S.J.: Feedback theory-some properties of signal flow graphs. Proc. Inst. Radio Eng. **41**, 1144–1156 (1953)
30. MATLAB. http://www.mathworks.com/products/matlab/. Accessed Aug 2012
31. Maybeck, P.S.: Stochastic Models, Estimation, and Control, Volume 1. Mathematics in Science and Engineering, vol. 141. Academic Press, New York (1979)
32. Murphy, K.P.: Dynamic Bayesian networks: Representation, inference and learning. Ph.D. thesis, University of California, Berkeley (2002)
33. Pearl, J.: Probabilistic Reasoning in Inteligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
34. Potts, C.: Software-engineering research revisited. IEEE Softw. **10**, 19–28 (1993)
35. Skaanning, C., Jensen, F.V., Kjærulff, U.: Printer troubleshooting using Bayesian networks. In: R. Logananthara, G. Palm, M. Ali (eds.) Intelligent Problem Solving. Methodologies and Approaches. Lecture Notes in Computer Science, vol. 1821, pp. 367–379. Springer, Heidelberg (2000)

Chapter 6
Supporting the Architecting Process of Adaptive Systems

Hitoshi Komoto, Roelof Hamberg, and Tetsuo Tomiyama

Abstract The creation of an adequate architecture is an essential ingredient to develop complex systems in an effective and economical way. In spite of many advances in model-based development, architecture creation still is more art than systematic engineering. This chapter describes a method and tool to support the architectural phase of the development of adaptive systems in a systematic and model-based way. In order to assist architects, the method discerns static decomposition and dynamic behaviours of the system, which are employed to structure the architects' work. The support encompasses amongst others the systematic generation of possible decomposition structures and the selection of the optimal subsystem decomposition by evaluating dynamic properties in a quantitative way. This method and tool together define and verify static decomposition and dynamic behaviours of the target system, and manage the interactions from various aspects across engineering disciplines. The method and tool are described and demonstrated with a case concerning the systems architecting part of a toner-based printer.

H. Komoto (✉)
Advanced Manufacturing Research Institute, National Institute of Advanced Industrial Science and Technology, Tsukuba, 305-8564 Ibaraki, Japan
e-mail: h.komoto@aist.go.jp

R. Hamberg
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: roelof.hamberg@esi.nl

T. Tomiyama
Department of BioMechanical Engineering, Faculty of Mechanical, Maritime, and Materials Engineering, Delft University of Technology, P.O. Box 5, 2600 GA Delft, The Netherlands
e-mail: t.tomiyama@tudelft.nl

6.1 Introduction

This chapter discusses the development of complex, adaptive, multi-disciplinary systems, such as professional printers. These systems typically involve an enormous number of components and details, involve a number of different disciplines, and require considerations over their entire life-cycle, which entails the involvement of various stakeholders. These factors increase the complexity of the product itself as well as of its development process.

Adaptivity[1] can be considered one of the ways to reduce and manage system complexities as perceived by its users. For instance, consider a printer that can automatically absorb fluctuations of paper thickness during operation – such self-managed changes are called adaptivity. Such a printer effectively reduces *operational complexity*,[2] because the system's user does not have to convey the value of the paper thickness parameter to the system. However, in this particular example, the adaptive printer will need additional components: a device to measure paper thickness, a device to physically change the pressure on paper, and control software to deal with these parameters at run-time. Obviously, in this case operational complexity is in trade-off with *design complexity*.[3] This is often true for adaptive systems, although sometimes a simpler design can lead to a simpler operation.

In order to manage design complexity, establishing a *systems architecture*[4] is considered a powerful method. In mechanical engineering, systems architecture (or *product architecture*) is often discussed in the context of modularity and product platform, focusing mostly on connectivity of components [4, 7, 12–15, 21, 30]. In software engineering, architecture relates to depicting how a system will behave, and serves as the primary carrier for system qualities and the definition of work assignments [31]. This chapter generalises these notions: systems architecture is used in its original sense that includes a variety of conceptual views ranging from functions to behaviours to connectivity of components.

In the literature (e.g. [3, 20, 24]), the development of the systems architecture (*systems architecting*) is defined in various ways. In this chapter we focus on a method and a computer-based tool to decompose system requirements to those of subsystems and components and to define their behaviours and interfaces, followed by a stage in which the best architectural solutions will be selected based on a set of performance criteria [17–19]. This method and tool reflect our goal to support the

[1] In this chapter the terms adaptivity and adaptability are used in a reciprocal manner.

[2] Operational complexity can be thought of as the amount of information needed to soundly operate the system.

[3] Design complexity can be thought of as the amount of information needed to soundly design the system.

[4] According to Wikipedia, *systems architecture* is the conceptual model that defines the structure, behaviour, and more views of a system.

early phases of development in a model-based way. Adaptivity must be embedded at the systems architecture level in this context.

In our approach aimed at the support for designing adaptive systems and their architecture, we identify two important elements. One is the *static decomposition* including structural organisation and connections of subsystems and components. The other is the *dynamic behaviour* (action plan) that describes how the components of the static decomposition operate together over time. These two elements interact with each other. To support their development we introduce a method and a tool. The computer-based tool is named SA-CAD (Systems Architecting Computer Aided Design). SA-CAD provides a systems architect with methods to conceptually break down top-level requirements into requirements at subsystem and component levels. Once this is done, it assists the architect to find feasible embodiments and to automatically generate possible decompositions. In this way, SA-CAD supports the systems architect to exhaustively search for the best static architectural solutions, minimising the chance of overlooking good design solutions. One of the best solutions will be selected by quantitatively evaluating dynamic behaviours (so-called action plans) with a systems dynamics model (for an external tool such as MATLAB/Simulink [22]) that SA-CAD builds from qualitative information.

The generality of the approach that we present here, relates this work in a natural way to other chapters of this book. The exploration of design candidates in order to come to a best architectural choice is also the topic of Chap. 7, where the application is geared towards data path design. The specification of the design across different views is a straightforward specialisation of the views addressed in this chapter. In Chap. 8, the connection with software architecting is made: the definition of behaviours coincides with their Domain-Specific Language (DSL) specification, which is the starting point for an automatically generated implementation. One of the central concepts of our method, the parameter network, is conceptually very close to the Bayesian networks of Chap. 5, which are introduced to deal in a model-based way with uncertainties. Chapters 3 and 4 on control approaches provide concrete solutions for system adaptivity.

This chapter is organised as follows. First, our approach to designing adaptive systems and their architecture is described. Next, the chapter explains our method for systems architecting and introduces the SA-CAD tool. The method and tool are demonstrated in the context of a case study concerning the systems architecting part of a toner-based printer.

6.2 Adaptive Systems: Architecture and Adaptability

Adaptive systems require architectures to cater for adaptability in several ways. In this section we illustrate how different forms of adaptability emerge in systems architecting.

6.2.1 Adaptability in the Literature

A variety of definitions of adaptability have been proposed in the literature. For example, Gu et al. distinguish *design adaptability* and *product adaptability* [11]. While design adaptability is an ability to adapt to new requirements by means of small design changes, product adaptability means a product that can be adapted by the user. According to Olewnik et al. [25, 26], an adaptable system is one of two types of flexible systems that retain "a high level of performance when operating conditions or requirements change in a predictable or unpredictable way". While a system that can accommodate unpredictable changes in an operating environment is called robust, one that can accommodate predictable changes is called adaptable. Some authors use "adaptability", "flexibility", and "reconfigurability" interchangeably. According to Rajan et al. [28], product flexibility is the degree of responsiveness for any future change in a product design while minimising the amount of effort to redesign. Ferguson et al. see adaptability as a means to achieve a reconfigurable system of which changeable variables as well as the range of their change are identified to improve the performance of the system, whereas a reconfigurable system is one that maintains a high-level performance by changing (in real-time) its configuration under changing operating conditions or requirements [8].

In this chapter we focus on self-governed adaptability of the product in which system parameters can be changed in a predictable manner to adapt to a new environmental situation or to improve the performance of the system. The adaptable system aims at extended utilisation of the system and its design. Extending the utility of the system can be performed at three different phases of its life cycle:

- During system design (design-time adaptability),
- During system operation (run-time adaptability); this results in so-called adaptive behaviour of the system, and
- During system service (adaptability for life-time or functionality extension).

These three types of adaptability are mutually interrelated, though. Design-time adaptability discusses how one design can be easily changed to another design and, thus, is deeply related to such concepts as evolvability [8, 36], product families [15], and upgradeability [35]. Life-time adaptability is relevant to upgradeability (or even downgradeability), retrofitability, and reconfigurability. This research particularly looks at run-time adaptability (adaptivity), which is more relevant in the context of designing the proper control mechanisms in such a way that they result in the desired system behaviour as a response to real-time changes in its own state and its environmental conditions.

Table 6.1 Interactions between architecture and behaviour

Strategy	Hardware	Circuit	Control	Interface
Adjustable	Adjustable fixers	–	–	Fixers
On-off	Mechanical sensors, actuators	Relay	Position	Positional sensors
Speed profile	Rotary encoder, actuators, PCB	Feedback circuit	Speed profile	Positional sensors, rotary encoder,
Regulatory control	Rotary encoder, actuators, MPU	MPU, AD-DA converters	Position, speed profile	Positional sensors, rotary encoder, AD-DA converters

6.2.2 *Adaptable Systems*

This chapter focuses on run-time adaptability and distinguishes two important elements of adaptable architectures, viz. *static decomposition* and *dynamic behaviour*. Static decomposition refers to the structural organisation and connections of subsystems and components, while dynamic behaviour (or action plan) describes how the components of the static decomposition operate together over time. These two elements of systems architectures interact with each other, which is relevant to consider during systems architecting.

As an illustrative example, suppose the distance of two physical components of a machine has to be changed (or controlled) during operation of that machine. Concretely, one may think of the distance between the rulers of a paper input tray of a printer that is meant to keep the paper at a well-defined position. A change of this distance may be necessary to perform a different task with the machine, for instance printing on different paper sizes. There exist several possibilities enabling this change, which we illustrate with four fundamentally different scenarios presented in Table 6.1. The realisation of the change in a scenario corresponds to the action plan that is used in the system. The four scenarios lead to different action plans. At the same time, depending on the scenario, the architecture will be completely different in terms of the configuration of components (sensors, actuators, circuits, and software), the control strategies, and possibly even the production methods. These interactions between static decomposition and dynamic behaviour must be correctly understood and captured in the systems architecting methodology.

The four scenarios of the example can be described as follows:

1. *Adjustable*. The two components will be fixed by means of adjustable fixers. This allows manual adjustment of the distances of the two components during assembly and operation.
2. *On-off*. One component is fixed and the other is driven by an actuator. When the distance reaches a preset value, the actuator stops. However, there is no control of the speed.

3. *Speed profile*. One component is fixed and the other is driven by an actuator. When the distance reaches a preset value, the actuator stops. The speed pattern of the moving component is predefined.
4. *Regulatory control*. One component is fixed and the other is driven by one or more actuators (even with multiple degrees of freedom). The motion (location, speed, acceleration) of the moving component must follow a pattern that is created at run-time.

For a concrete system, one would choose one of the possible scenarios, depending on the required system properties. While the first scenario provides a non-adaptive, simple, and low-cost solution, the other scenarios each provide an increasing level of adaptivity that may be desirable in terms of usability and productivity at a high cost level.

The following sections illustrate a systems architecting method and subsequently a computer-based tool called SA-CAD. This method and tool together can define and verify static decomposition as well as dynamic behaviours of the target system, and manage the interactions that exist between all the different views.

6.3 Supporting the Systems Architecting Process

In order to support the systems architecting process, the characteristics of the process have to be analysed. Subsequently, the needs for supporting this process in a better way have to be made explicit. This section deals with these two topics.

6.3.1 The Systems Architecting Process

The systems architecting process of mechatronic systems defines the physical and logical configuration (organisation) of subsystems and components that realise desired functions and behaviours. It is largely defined through decomposition of system-level specifications to specifications of subsystems and components [17]. In the systems architecting process, the hierarchy of systems is observed from functional and physical aspects as well as the organisation of a design team.

The V-model (see Fig. 6.1) has been proposed as a framework of product development projects considering hierarchical system decomposition and integration [10]. In the literature [6, 29, 37] it is argued that the V-model is particularly suited for an architecting process that deals with systems having hierarchical structures for a range of various aspects. The V-model consists of three main phases, i.e. the system decomposition phase, the implementation phase, and the system integration and verification/validation phase. The special focus of the system decomposition phase is the translation of high-level requirements and constraints into design solutions. The architecting activity discussed in this chapter is performed

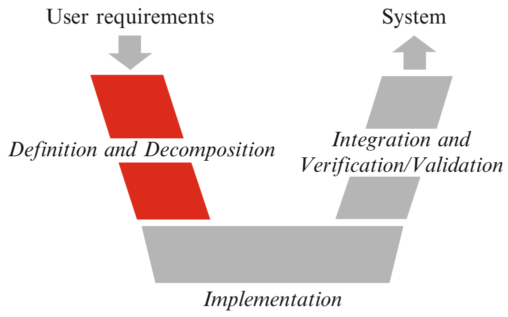

User requirements
System
Definition and Decomposition
Integration and Verification/Validation
Implementation

Fig. 6.1 Product development based on the V-model [10]

in this phase. In general, the architecting process consists of the following fundamental tasks [19]:

- Identification of requirements to be translated into the system-level specifications (i.e. functions) of systems.
- Hierarchical decomposition of the system-level specifications to the specifications of subsystems and eventually to components, taking technological realisation constraints into account.
- Definition of behaviours and structure of these subsystems and their interfaces to meet the corresponding specifications, taking constraints of physical laws of nature and possible control behaviours as determined by chosen technologies into account.

After these tasks have been completed, components and subsystems are designed in detail (this can be considered part of the implementation phase). Finally, validation and verification of designed subsystems and components are performed with their integration.

6.3.2 Requirements for Systems Architecting Support

The main requirements of methods and tools to support the systems architecting process of adaptive systems are listed below:

- *Multi-disciplinary concurrent design*
  Concurrent design decreases lead-time of product development through collaboration of design teams. It can be compared with classical development processes of mechatronic systems, which start with mechanical design followed by the design of control of sensors and actuators, and software requiring experts from various engineering domains. In systems architecting, collaboration of these domain experts is necessary, within a design team as well as between design teams. Furthermore, systems architects organising collaboration between domain experts are necessary.

- *Systematic design*
  Systematic design [27] enables systematic generation and exploration of candidate design solutions, which decreases the chance of overlooking good design solutions as well as the chance of encountering crucial design faults. In designing complex mechatronic systems, crucial design faults are typically discovered at the time of integration of subsystems [33], while they are not observed during the design of individual subsystems. In systems architecting, potential design faults regarding integration of subsystems should be detected and avoided [5].
- *Model-based design*
  Commercial systems engineering tools (e.g. MATLAB/Simulink [22], Modelica [23], 20-sim [1], and numerous others) increasingly support model-based design of mechatronic systems across the boundary of engineering domains. Such support includes multi-phase and multi-physics simulations. These tools support the development of computable and executable system models, which are used in later phases of system development. Methods and tools to support the systems architecting process, which takes place prior to these phases, should increase the degree of automation of the generation process of such system models as well as their management.

6.4 A Support Tool for Architects: SA-CAD

To support architects of adaptive mechatronic systems, we have developed a computational tool called SA-CAD (Systems Architecting CAD) [19]. The main features provided by SA-CAD are listed below, while their details are described later.

- *Multi-disciplinary concurrent design*
  A model made in SA-CAD is independent from any specific engineering domain language. Especially, SA-CAD employs graphical definitions of structure and behaviour connected with both abstract and parameter-level descriptions. These definitions allow architects to model a product with a simple geometric modeller (structure) and a process modeller (behaviour). SA-CAD also manages consistency between models from different aspects.
- *Systematic design*
  SA-CAD supports systematic generation of alternative design concepts and the baseline evaluation of the performance of generated concepts. The support is realised with a qualitative product model in SA-CAD or quantitative product models in external analysis tools. This separation of model support is motivated by the large number of available tools for performing quantitative analyses.
- *Model-based design*
  SA-CAD allows architects to model a product with abstract (symbolic) descriptions as well as parameter-level descriptions. SA-CAD exports quantitative product models that can be used for systematic evaluation of design concepts.

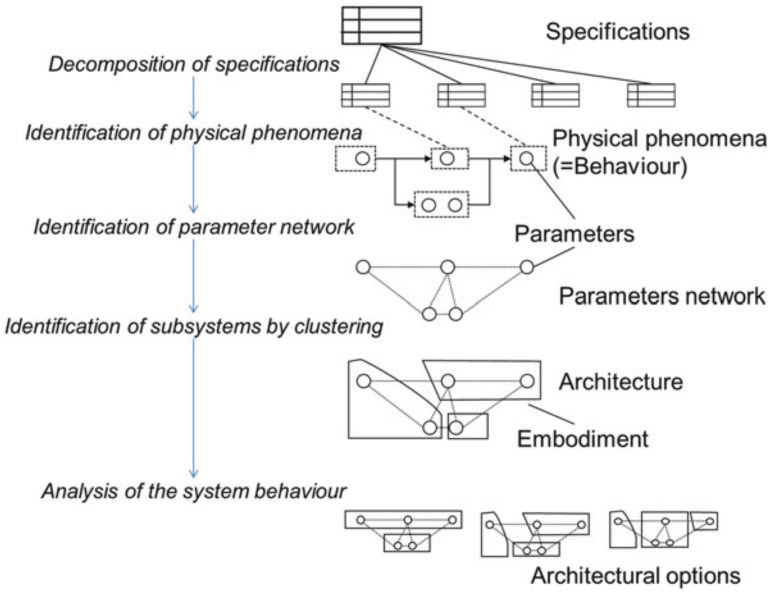

Specifications
Decomposition of specifications
Identification of physical phenomena
Physical phenomena (=Behaviour)
Identification of parameter network
Parameters
Parameters network
Identification of subsystems by clustering
Architecture
Embodiment
Analysis of the system behaviour
Architectural options

Fig. 6.2 The systems architecting process with SA-CAD

6.4.1 The Systems Architecting Process with SA-CAD

The aforementioned features of the SA-CAD tool can be applied in the systems architecting process shown in Fig. 6.2. We claim that this process, which fits our methodology, is a quite natural formalisation of the architect's tasks when developing a systems architecture.

According to this process, an architect first defines the high-level specifications of systems such as functional requirements and technical performance indicators. They are derived from user requirements. For instance, the architect analyses the need of users to "print with a high speed on heavy media on both sides" and translates this into technical performance indicators such as "the admitted range of media mass" and "the maximum engine speed", while the description "to print on both sides" is treated as a functional requirement.

Subsequently, the architect searches for subsystems, such as mechanisms, sensing devices, and software components. These subsystems should be integrated in such a way that the system can satisfy the high-level specifications. In the course of this search and integration, high-level specifications are decomposed into low-level specifications for subsystems, which can be budgeted and measured as independently as possible. For instance, the total size of a printer restricts the size of subsystems. At the same time, the total energy consumption constrains the energy consumption of each subsystem.

At the decomposition of high-level specifications, the architect may not be able to readily find appropriate subsystems from past designs. In this case, he/she will continue the decomposition process and define lower-level specifications, until the level at which he/she can identify desired behaviours to meet these lower-level specifications. The desired behaviours are defined in terms of physical phenomena and processes, which include parameters that will form a network. This network of parameters plays a crucial role in system decomposition and systems architecting.

Subsystems do not correspond one-to-one to behaviours. There are subsystems realising multiple behaviours, and behaviours involving multiple subsystems. Relations between behaviours and subsystems are defined in terms of the parameters of a system. These parameters are derived from behaviours, and form a network that can be divided into clusters. The network represents a system and a cluster represents a subsystem (or embodiment of the subsystem). The design of complex systems is not only defined by the parameters of systems related to the behaviours, but also by the clustering patterns of the parameters.

A set of clusters represents a decomposition of a system or an architecture. Other sets of clusters derived from the same parameters are alternative candidates of the decompositions of the system. In our methodology and supported by our tool, the architect systematically generates and evaluates possible architectural decomposition candidates. To evaluate a decomposition candidate, de facto he/she analyses the parameter network of the candidate. The analysis may be based on qualitative relations between the parameters, or quantitative relationships that are derived from the detailed parameter relations (e.g. differential equations). The architect may employ external tools to analyse such relationships. For instance, the information flow in a printer may be analysed by relations between the parameters of sensors and actuators, while the temperature of subsystems in a printer may be numerically analysed by a finite element method.

6.4.2 The Concept of SA-CAD

SA-CAD supports the systems architecting process that we introduced in the previous subsection. Figure 6.3 depicts the conceptual overview of SA-CAD. To model a system using SA-CAD, the architect can use four aspect modellers: an *FBS modeller* (Function-Behaviour-State modelling for a qualitative model of the system consisting of functional, behavioural, and structural components) [34], a simple *geometric modeller* (for quantitative structure support), a *parameter network modeller* (for tracking and tracing parameter relationships), and a *process modeller* (for quantitative behaviour support). SA-CAD further provides modelling support through knowledge bases and reasoning algorithms. Knowledge bases contain physical concepts and laws that describe known behaviour [38], while reasoning algorithms can generate candidate solutions for given functional specifications.

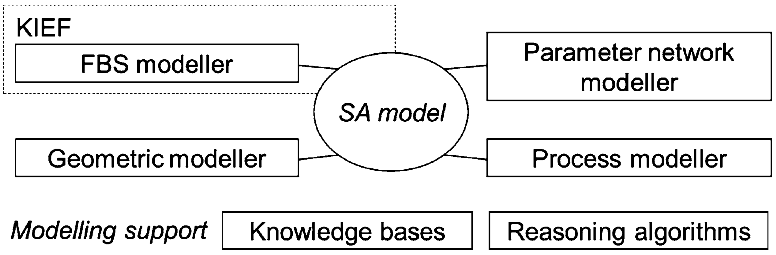


Fig. 6.3 Conceptual overview of SA-CAD [19]. KIEF denotes knowledge intensive engineering framework [38]

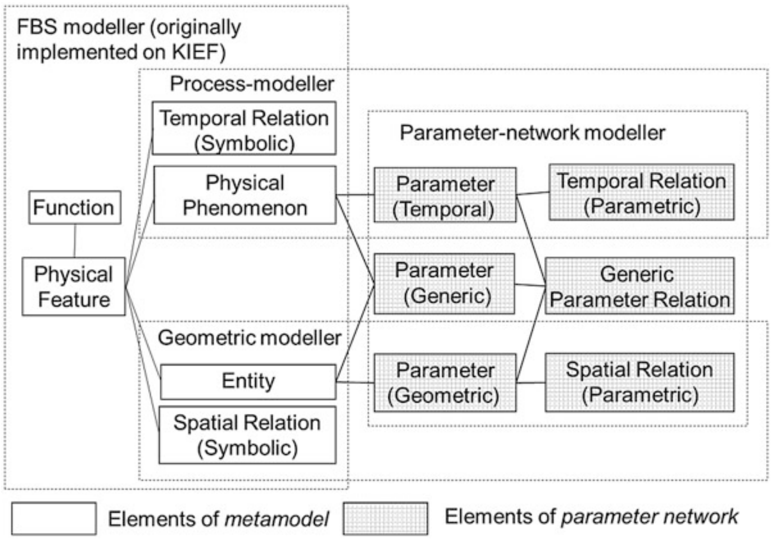

FBS modeller (originally implemented on KIEF)
Process-modeller
Temporal Relation (Symbolic)
Physical Phenomenon
Function
Physical Feature
Parameter-network modeller
Parameter (Temporal)
Temporal Relation (Parametric)
Parameter (Generic)
Generic Parameter Relation
Geometric modeller
Entity
Spatial Relation (Symbolic)
Parameter (Geometric)
Spatial Relation (Parametric)
Elements of metamodel
Elements of parameter network

Fig. 6.4 System modelling using SA-CAD. The *FBS modeller* supports the development of a product concept in terms of functions and physical features. A physical feature includes a set of symbolic descriptions of the behaviour and structure in terms of physical phenomena and entities. Constraints on physical features are defined symbolically with temporal relations between physical phenomena and spatial relations between entities. The *parameter network modeller* supports the development of the parameter network of a product, which represents the parameters and their relations characterising its detailed behaviour and structure. The *geometric modeller* and the *process modeller* support visualisation of the behaviour and structure of a product. The relations between these parameters are computed and evaluated in terms of consistency of the symbolic and parametric descriptions of the behaviour and structure of a product. Details can be found in [19] and are illustrated in the printer case study (Sect. 6.5)

Figure 6.4 shows the model of a system in SA-CAD and the responsibility of each aspect modeller for the model development. The details of the concept of SA-CAD are explained in [19].

A system model developed with SA-CAD, or an SA-model, basically consists of an FBS model and a parameter network. The FBS model is the conceptual network of the elements of a system concerning its functions, behaviours, and structures; this model is developed with the FBS modeller. The elements include functions (top-level specifications, with subdivisions into lower-level functions), physical phenomena (originating from laws representing natural phenomena as well as designed-in control processes), and entities (subsystems that implement one or more functions), which are together capable of representing part of the knowledge in the systems architecting process shown in Fig. 6.2. Relations between the model elements in terms of behaviours (e.g. the order of process activation) and structures (e.g. the hierarchy of entities) are defined in terms of temporal relations between physical phenomena and spatial relations between entities.

The parameter network, which is developed in the parameter network modeller, represents parameters and their relations characterising its detailed behaviour and structure. To this end, these parameters are connected with the physical phenomena and entities in the FBS model. In order to describe the behaviours and structures with the process modeller and the geometric modeller, the geometric and temporal parameters store the quantitative information of the entities and physical phenomena in the FBS model. The temporal and geometric relations in the FBS model are used to check the consistency of the model with the geometric modeller and the process modeller.

SA-CAD's development is based on the earlier versions of the Knowledge Intensive Engineering Framework (KIEF) [38], which supports conceptual design processes based on FBS modelling. The knowledge bases and reasoning algorithms used in KIEF are also used by SA-CAD, although the explanation and demonstration of these algorithms are beyond the scope of this chapter. The algorithms include:

- Qualitative Process Abduction System (QPAS) [34],
- Physical Feature-based Reasoning System (PFRS) [38] for development, and
- A qualitative reasoning system of the behaviour of a product in terms of parameter value changes based on Qualitative Process Theory (QPT) [9] for verification.

6.5 A Printer Case Study

This section illustrates the usage of SA-CAD in the architecting process of a printer. The architecting process is complex in terms of the number of subsystems and the number of multi-disciplinary interrelations between the subsystems. SA-CAD provides architects with the support explained in the previous sections. Along the steps of the SA-CAD architecting process shown in Fig. 6.2, we illustrate the activities in the context of the case study. The details of this case study on the topics of "development and analysis of a qualitative printer model" and "systematic generation of architectural options" can be found in extended publications ([17] and [18], respectively).

6.5.1 Knowledge Mining for Systems Architecting

Before and during the architecting process of a printer with SA-CAD, architects and domain experts continuously collect knowledge of printers and their subsystems, which can be used in the architecting process. It is a combination of what is to be achieved with the printer and how this can be done, based on experiences gained in the development of previous products, matched with the requirements of the new system under development. This knowledge is categorised into a few categories, discussed in the following sections.

6.5.1.1 Function Knowledge

Function knowledge represents the overview of functions of a product (a printer in this case study). This knowledge is generally rather stable over a range of products in a specific domain. It represents the domain knowledge of how the product functionality is ultimately realised.

Function knowledge is described with *to do something*, i.e. a pair of a verb and a noun that together fulfil an objective [34]. Other researchers describe function knowledge with inputs and outputs of material, energy, and information (cf. [32]). Function knowledge can be hierarchically structured. The resulting hierarchical structure is called functional decomposition or function hierarchy, which is used to relate physical phenomena and entities that realise functions. The hierarchy can be detailed in order to organise the knowledge about the detailed behaviours and structures. A rough example is the breakdown of the function *print a sheet* into functions *transport a sheet* and *transfer a toner image*, which relate to the entity *concrete paper path* with *controlled speed of the sheet* as behaviour, and the entity *belt-based toner transferrer* that behaves properly when *the toner image is completely moved from belt to sheet*.

6.5.1.2 Structural and Behavioural Knowledge

Architects also collect structural and behavioural knowledge, which is ultimately applied to realise functions. It is more specific knowledge about the product realisation itself, that is closely related to choices made in the organisational context. Nevertheless, the knowledge is based on a wealth of properties of entities and/or materials that are just given by nature itself.

This category of knowledge consists of a set of entities, physical phenomena (and processes) that occur with entities, temporal relations between physical phenomena, spatial relations between entities, and parameters (attributes) that characterise entities. Architects collect the knowledge by analysing the structure and behaviour of existing printers, or, alternatively, by performing new experiments in the context of technology development. Architects cooperate with domain experts (such as mechanical, control, and software engineers), because the major part of the

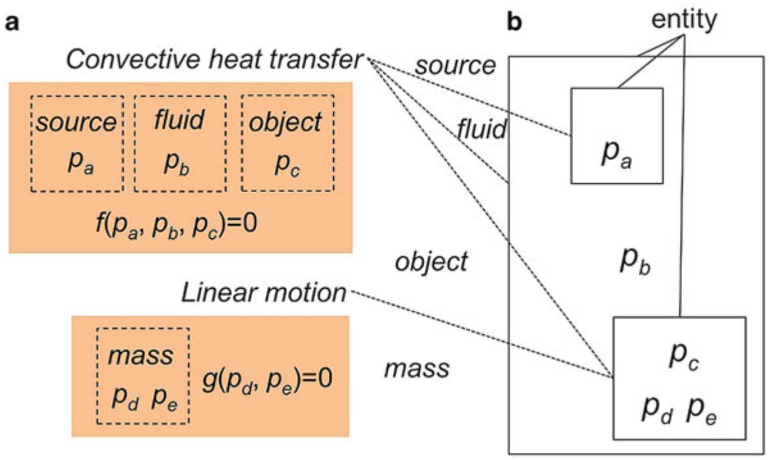


Fig. 6.5 Examples of qualitative parameter relations. In (**a**), two physical phenomena are modelled as building blocks with two laws f and g, involving entities (*source*, *fluid*, *object*, *mass*) and parameters (p_a to p_e). In (**b**), a system structure involving three entities and two phenomena is modelled. In the systems architecting process, object and mass, which are separately defined in the physical phenomena, are regarded as one single entity. In our case, this could reflect a moving sheet that is heated by hot air coming from a heater. The *dotted lines* indicate connections to phenomena, while *solid lines* indicate entities

knowledge is developed and maintained by these domain experts. Architects use this knowledge to be able to better balance choices across product functions and disciplines needed to realise these functions.

The structural and behavioural knowledge is defined in terms of parameters and their relations. These parameters are classified into, for instance, constants and variables, observable parameters and controllable parameters, design parameters and performance targets. Relations between parameters indicate, for instance, inputs and outputs of software and control processes, and relations and influences between parameters at the occurrence of physical phenomena. The applicability of the knowledge requires it to have a quantitative nature, i.e. with typical values for parameters, concrete processes, etc.; this makes such knowledge domain-specific.

In SA-CAD, the parameters and their relations are stored as *building blocks* of physical phenomena. Figure 6.5a shows a building block made of two physical phenomena. In Fig. 6.5a, *convective heat transfer* is a physical phenomenon, which is described by heat flow from *source* to *object* through *fluid*. The physical phenomenon is further described by some parameters and their relations such as the quantity of heat flow depending on the temperature of *object* and *source*, and the thermal conductivity of *fluid*. Here, source, object, and fluid are all entities that are part of the building block. Building blocks are independent of the system's model itself (much alike the relation between class and object in object-oriented software). Similarly, *linear motion* is defined with a set of parameters (e.g. mass and acceleration) of a moving object *mass*. Figure 6.5b shows a model with three

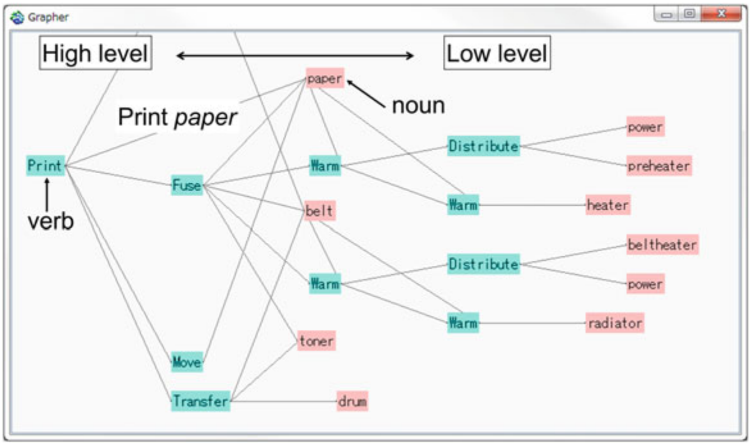


Fig. 6.6 FBS model from a functional view (adapted from [17]). Any combination of a *blue* verb with a *pink* noun is a function, which together are organised in a hierarchy

entities and both physical phenomena that are applicable to them. One of the entities is regarded as *object* of *connective heat transfer* and the *mass* of *linear motion* (i.e. an entity that absorbs heat and moves, respectively). This entity inherits the parameters of *object* and *mass* through its connection to the two phenomena.

6.5.2 *Development and Analysis of a Qualitative Printer Model*

After the preparation of the knowledge explained in the previous subsection, the architect develops a printer model in SA-CAD with this knowledge in order to support the development of an actual printer. The focus of this subsection in consecutive parts is on the FBS modelling of the printer, graphical definition of the structures and processes, consistency evaluation of the developed FBS model with the graphical definition, and the analysis of the parameter relations derived from the FBS model.

6.5.2.1 FBS Modelling

The architect starts with defining an FBS model of the printer using the FBS modeller in SA-CAD. This model explains how the functional decomposition is realised by structures and resulting system behaviour.

The computational support of this step includes function modelling and reasoning, and design synthesis [34]. Figure 6.6 shows an FBS model from a

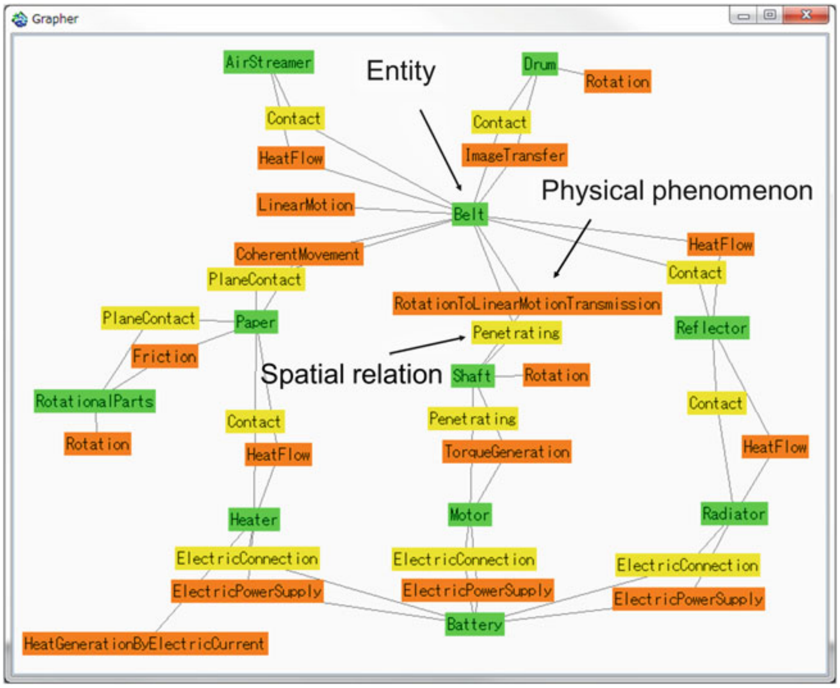
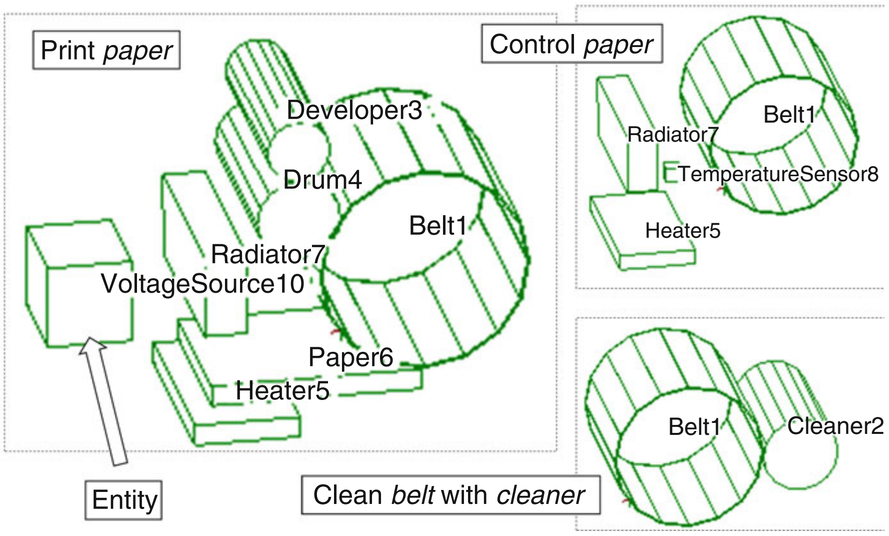


Fig. 6.7 FBS model from a structural and behavioural view. *Green* system entities, *orange* phenomena, and *yellow* spatial relations together form a network of relationships

functional view, which is the overview of the printer developed through a functional decomposition. Functions are described by (blue) verbs with (pink) nouns. Nouns in Fig. 6.6 are used to characterise verbs to help the architect understand the required concepts of the printer from the functional view. Each bottom-level function has a realisation defined in terms of physical phenomena, entities, and relations between entities. Figure 6.7 shows the FBS model from a structural and behavioural view with a graph representation. Nodes in the graph are (orange) physical phenomena, (green) entities, and (yellow) spatial relations. The graph is made of a set of physical features corresponding to low-level functions. During FBS modelling, physical features are instantiated after functional decomposition. Entities (e.g. *Paper* and *Belt*) defined across multiple physical features are unified in order to define the concrete entities in this specific printer case, which are subject to multiple physical phenomena defined in various physical features. This unification is a common process in the FBS modelling with a knowledge base, in which the design knowledge is organised and indexed in terms of functions. Note, however, that the embodiment with physical features at this stage (e.g. Paper and Belt) is not yet the final architecture. Section 6.5.3 describes the final architecture determined after clustering parameters.



object1	object2	FBS	Geometric Model	Evaluation
Cleaner2	Belt1	Contact	PlaneSupported	consistent
Belt1	Heater5	not defined	PlaneSupporting	suggestion for addition
Belt1	TemperatureSensor8	Contact	NoContact	not consistent
Drum4	Belt1	Contact	PlaneSupported	consistent
Belt1	Developer3	not defined	PlaneSupporting	suggestion for addition

Fig. 6.8 Geometric views and the results of consistency analysis (adapted from [17])

6.5.2.2 Graphical Definition of Geometry and Process

Two additional modellers are introduced here to enable the architect to specify constraints in certain views. The physical layout and the operational behaviour are important examples of such views, dealing with relations in space and time, respectively.

The geometric modeller imports the entities defined in the FBS model. SA-CAD uses functions in the FBS model (see Fig. 6.6) to specify the scope of information displayed on the modeller. Within a specific (relevant) scope, the architect can define the detailed geometric information using the modeller. For example, Fig. 6.8 shows several geometric views corresponding to intermediate functions (of the function hierarchy): "*print* paper", "*clean* belt with cleaner", and "*control* paper", all taken from Fig. 6.6. In line with the selection of functions, some entities are not displayed in the modeller. For instance, cleaner and temperature sensor are not necessary to realise the printing function, unless the control and cleaning of the printer are considered. The graphical definition of an entity includes the primitive shape (e.g. cube and cylinder) and the parameters to specify the size (scale) and the relative position (translation and rotation). The architect defines the values of

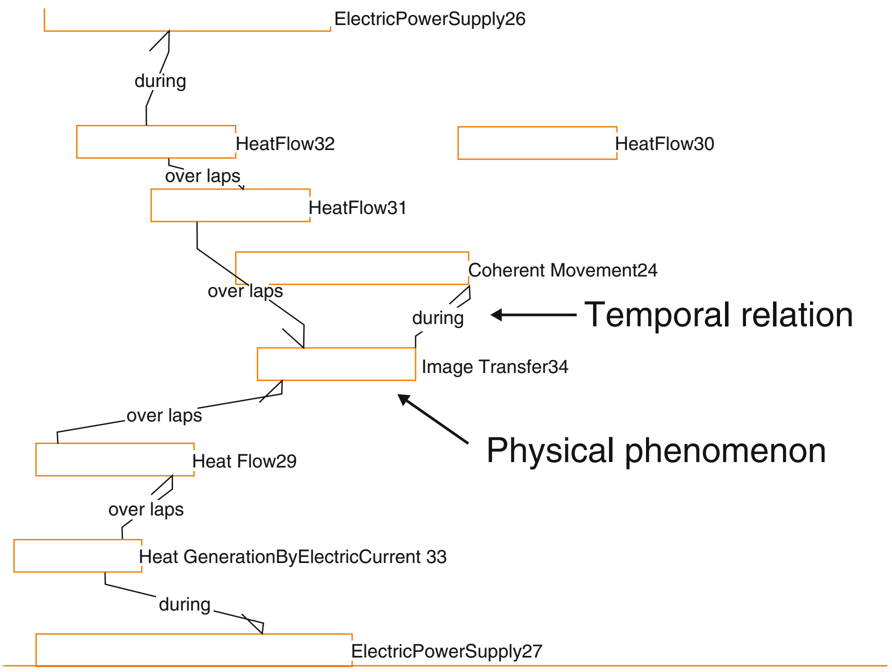


Fig. 6.9 Process view in which temporal relationships between different *orange* phenomena are modelled

these parameters with the modeller. The table at the bottom of Fig. 6.8 shows the evaluation of a consistency check with the FBS model, which is further explained in the next section.

Similarly, the process modeller imports the physical phenomena defined in the FBS model. With the modeller, the architect defines the occurrence and termination of physical phenomena with respect to time, and the symbolic temporal relations between them. The temporal relations are formally defined with Allen's interval temporal logic [2]. This logic defines 13 basic relationships between any two pairs of activities over time: *before*, *after*, *during*, *contains*, *overlaps/overlapped-by*, *meets/met-by*, *starts/started-by*, *finishes/finished-by*, and *equals*. These basic relations can be used to specify temporal relations between any two processes. Figure 6.9 shows the processes and their temporal relations. These processes are made of toner image transfer processes and individual heat flows that are temporally constrained by the toner image transfer processes. The timing information about each process is represented by the position of these processes with respect to the horizontal time-axis. These processes correspond to toner transfer functions and heating functions. The temporal condition of the toner transfer function in Fig. 6.9 is interpreted as "image transfer from drum to belt occurs during coherent movement between belt and drum". The temporal condition of the heat transfer processes

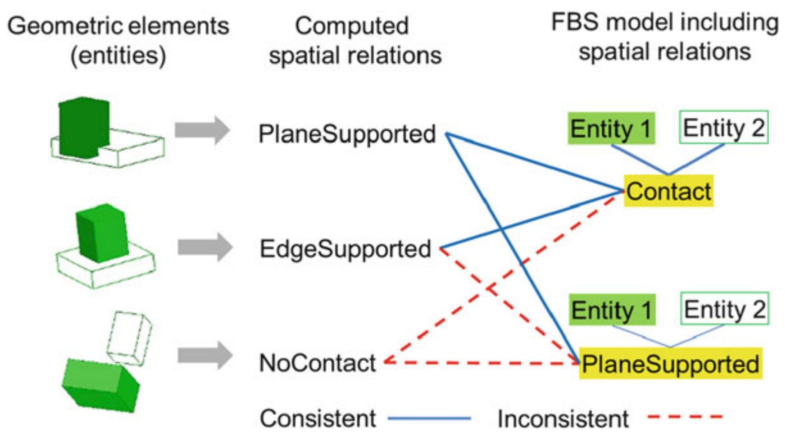


Fig. 6.10 Structural consistency evaluation with SA-CAD. Some spatial relations are consistent with the specified geometric relations in the FBS model, others are not

indicates that "heat generation occurs to multiple locations (i.e. multiple heaters) and individual heat flows start before image transfer from drum to belt starts". To define the toner image transfer process, the architect cooperates with a mechanical engineer and a software engineer who separately develop the physical structure and the control software, respectively.

6.5.2.3 Consistency Evaluation

The architect can evaluate consistency between the symbolic spatial relations defined with the FBS modeller and the geometric information defined with the geometric modeller. The evaluation procedure is illustrated in Fig. 6.10. SA-CAD computes quantitative relations between two geometric elements and outputs symbolic spatial relations. The computation applies Allen's interval logic[5] [2] to each pair of Cartesian coordinates. Subsequently, SA-CAD compares the computed symbolic spatial relations with the spatial relations defined in the FBS model. The procedure to calculate the symbolic spatial relation between two geometric elements and the concept hierarchy of the symbols of spatial relations are shown in [16].

Figure 6.8 showed already a result of the comparison in a table format. The spatial relations defined in the FBS model are shown in the third column labelled "FBS", while those calculated based on the geometric information are shown in the fourth column labelled "Geometric Model". The fifth column, labelled "Evaluation", shows the result of comparison, which is explained next. For example, the second row in the result shows that the spatial relation between *Cleaner* and *Belt*

[5]The interval logic applies to temporal relations as well as to spatial relations.

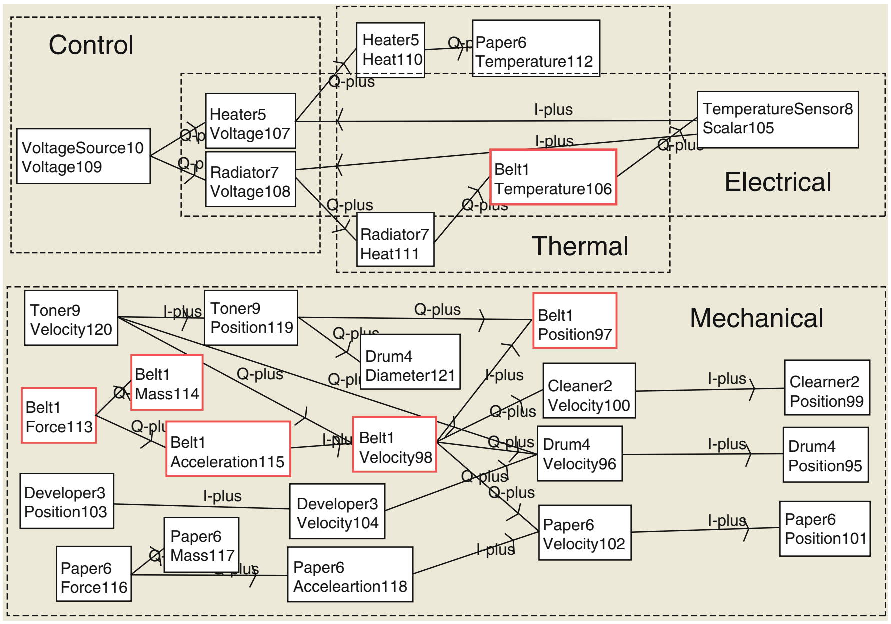


Fig. 6.11 Parameter network (in the parameter network modeller) (adapted from [17])

is consistent. This means that the relation defined with the FBS model, which an architect adds as a requirement to realise the function *clean belt*, is consistent with the geometric information defined with the geometric modeller by another architect or a mechanical engineer. SA-CAD calculates the relation as *PlaneSupported*, which is a subclass of *Contact* [16]. SA-CAD also suggests addition of symbolic spatial relations when some contact relations are calculated based on the geometric information defined with the geometric modeller, but not explicitly defined in the FBS model. In this case, *suggestion for addition* is displayed as the result of evaluation.

Such a consistency evaluation is similarly applied to the behavioural domain (i.e. with temporal relations between processes). The consistency evaluation is performed at any level of functional decomposition. Therefore, SA-CAD can identify inconsistent descriptions at a higher function level, which are consistent in the scope of the lower function levels.

6.5.2.4 Parameter Relation Analysis across Engineering Domains

In this case study, SA-CAD helps the architect to analyse dependencies between the parameters of a printer. The analysis is performed with the parameter network modeller, which displays a dependency network across engineering domains. Figure 6.11 shows the parameters of a printer and their dependencies, including mechanical, thermodynamic, electrical, and control domains. Figure 6.11 indicates that some entities have parameters in multiple domains (e.g. see the emphasised parameters of *Belt* in Fig. 6.11), and that parameters in the mechanical domain are isolated from those of the other domains. The dependencies are defined in terms of qualitative relations defined for qualitative simulation based on Qualitative Process Theory [9]. In Fig. 6.11, "Q-plus" indicates a qualitative proportional relation between two parameters (e.g. the *voltage* supplied to *Radiator* and the generated *heat*), while "I-plus" indicates a qualitative differential relation between two parameters (e.g. the *velocity* and *position* of *Belt*). Observation of such dependencies helps the architect to consider and express the introduction of physical phenomena and processes for increasing the adaptability of a printer. For instance, specific to the case in Fig. 6.11, the architect can consider and express the possibility to modify the *velocity* of *Belt (98)* by measuring the *temperature* of *Paper (112)* by adding a dependency between two parameters in the parameter network modeller.

The architect defines these parameters and their dependencies with the other three modellers, whilst the analysis is performed with the parameter network modeller. This means that the dependency network in Fig. 6.11 is generated in an automated way. The ingredients come from the FBS model. Further, some of the parameters and their dependencies are defined in the knowledge base of physical phenomena. Finally, the geometric modeller and the process modeller specify spatial and temporal parameters, respectively.

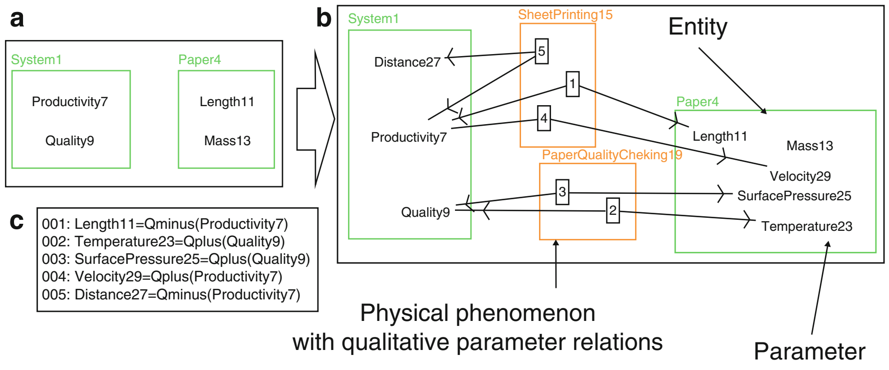
6.5.3 Systematic Generation of Architectural Options

SA-CAD supports the generation of architectural options (i.e. candidates of physical decompositions) in the architecting process. It does this as part of the following interactive procedure that is iteratively executed in the systems architecting process:

1. The architect defines the target parameters.
2. SA-CAD displays the physical phenomena that can influence the target parameters.
3. The architect selects at least one of the physical phenomena for each of the target parameters.
4. SA-CAD displays the available architectural options.

The generation process is performed by the parameter network modeller, which also displays the mapping of parameters to entities and an explicit description of the parameter relations. An example of the decomposition procedure is given below.

Consider a simple example in which, before conducting the above four steps, the architect builds an initial model that consists of two entities, *printer* and *paper*, and four parameters, *quality* and *productivity* belonging to printer and *length* and *mass* belonging to paper. Figure 6.12a shows the initial parameter network model in which the boxes are entities containing parameters. These parameters represent



001: Length11=Qminus(Productivity7)
002: Temperature23=Qplus(Quality9)
003: SurfacePressure25=Qplus(Quality9)
004: Velocity29=Qplus(Productivity7)
005: Distance27=Qminus(Productivity7)

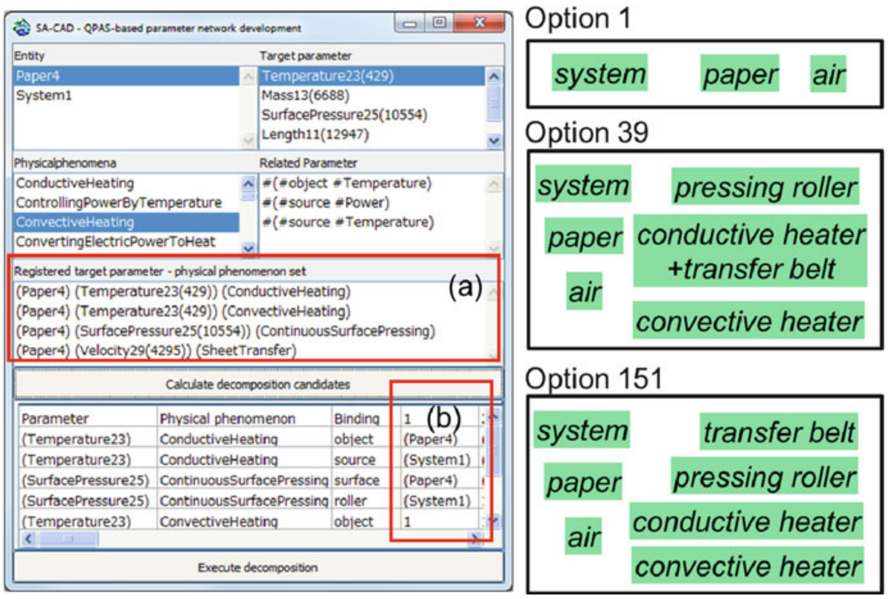
Fig. 6.12 Identification of system parameters based on customer requirements (adapted from [18])

customer requirements about the performance and operational range of the printer, i.e. system-level input and output parameters.

In Step 1, the architect defines *quality* and *productivity* as target parameters. In Step 2, SA-CAD finds physical phenomena or processes, which relate parameters of the printer with these target parameters. In this case, SA-CAD finds and displays a process *sheet printing* defined in its own knowledge base, because the process includes parameter relations indicating that *productivity* is proportional to the *velocity* of paper, inversely proportional to the *length* of paper and the *distance* between sheets (as a parameter of the printer). In Step 3, the architect selects the process in order to add the process and parameters to the model. There are no architectural options at this moment (i.e. Step 4 is omitted in this iteration), because the added processes do not introduce new entities. Figure 6.12b shows the updated model, in which physical phenomena (processes) containing the indices of parameter relations are displayed. Figure 6.12c shows the details of the parameter relations corresponding to the indices in Fig. 6.12b. The main architecture of the printer is determined through the entire four steps listed at the beginning of this section.

A number of iterations later, the left hand side of Fig. 6.13 shows the screenshot of the system decomposition interface. Starting again with Step 1, *temperature*, *velocity*, and *surface pressure* of *paper* are selected as target parameters. In Step 2, SA-CAD suggests several candidate physical phenomena that can potentially change the value of the target parameters. For instance, *convective heating* and *conductive heating* are suggested as physical phenomena regarding *temperature* of *paper*. In Step 3, the architect selects four physical phenomena in order to change the target parameters. Figure 6.13a in the interface shows the list of selected physical phenomena and the corresponding target parameters.

In the last part of this iteration, in Step 4, the tool suggests 151 architectural options taking the aforementioned list as input. The right-hand side of Fig. 6.13 shows three architectural options, in which entities that are potentially instantiated as new entities are named (e.g. *transfer belt* and *conductive heater*). Architectural



Entity	Target parameter
Paper4	Temperature23(429)
System1	Mass13(6688)
	SurfacePressure25(10554)
	Length11(12947)

Physicalphenomena	Related Parameter
ConductiveHeating	#(#object #Temperature)
ControllingPowerByTemperature	#(#source #Power)
ConvectiveHeating	#(#source #Temperature)
ConvertingElectricPowerToHeat	

Registered target parameter - physical phenomenon set
(Paper4) (Temperature23(429)) (ConductiveHeating)
(Paper4) (Temperature23(429)) (ConvectiveHeating)
(Paper4) (SurfacePressure25(10554)) (ContinuousSurfacePressing)
(Paper4) (Velocity29(4295)) (SheetTransfer)

Parameter	Physical phenomenon	Binding	1
(Temperature23)	ConductiveHeating	object	(Paper4)
(Temperature23)	ConductiveHeating	source	(System1)
(SurfacePressure25)	ContinuousSurfacePressing	surface	(Paper4)
(SurfacePressure25)	ContinuousSurfacePressing	roller	(System1)
(Temperature23)	ConvectiveHeating	object	1

Fig. 6.13 System decomposition interface and examples of suggested decomposition candidates (adapted from [18])

option 1 shows the decomposition in which a minimum number of new entities are added to the model. Architectural option 151 shows the decomposition with the maximum number of entities added to the model. In this decomposition, all potential entities, which do not correspond to paper, are treated as individual entities. Architectural option 39 shows a decomposition in which two entities (*conductive heater* and *transfer belt*) are treated as a unified entity. Figure 6.14 shows architectural option 39 after its selection by the architect.

The algorithm to identify physical phenomena to influence the value of target parameters is based on the algorithm implemented in the Qualitative Process Abduction System (QPAS) (see [18, 34]).

6.5.4 Qualitative Model Analysis with an External Tool

In order to support quantitative analyses, SA-CAD was extended to support linking external tools to it. In this concrete case, SA-CAD supports the architect to analyse architectural options in terms of parameter relations with MATLAB/Simulink [22], which represents a model with input-output representation of signals, and simulates the model behaviour. The analysis is useful to understand the behaviour of a printer (e.g. dynamics of the temperature of subsystems and power distribution) under

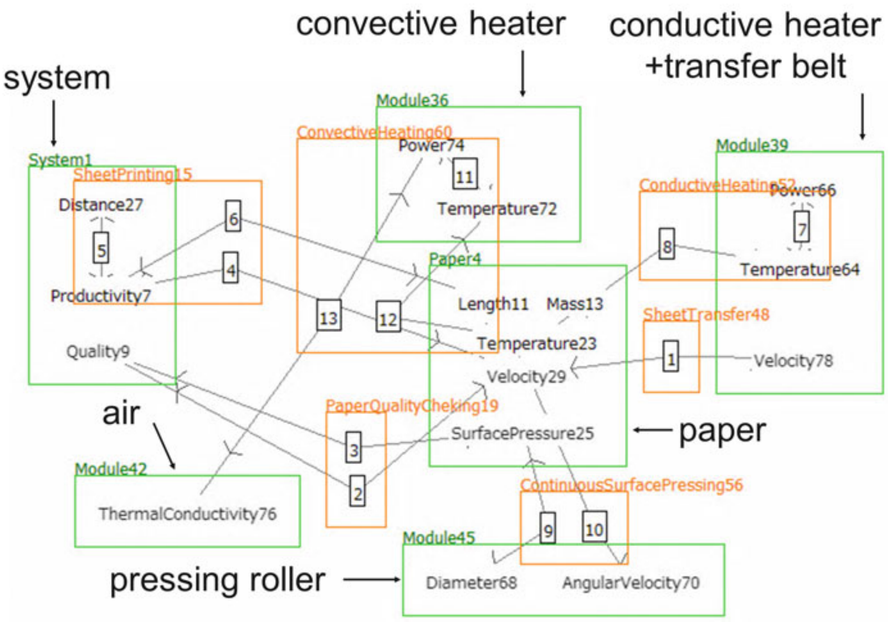
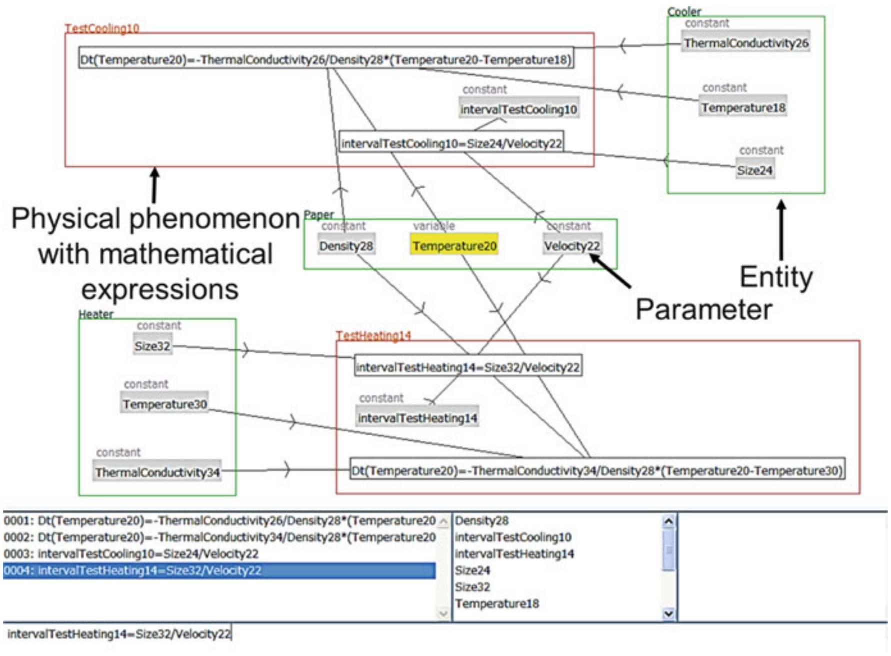


Fig. 6.14 A system decomposition result [18]. The *arrows* together with the indices indicate phenomena. The shown result is generated in an automated way, although the layout of the figure was elucidated by hand

different parameter settings (e.g. external temperature and variation of paper mass). In addition to the function of SA-CAD to analyse parameter dependencies as shown in the previous subsection, such analysis is crucial for the detailed design of adaptive systems.

For the analysis with MATLAB/Simulink, SA-CAD exports information required for this analysis. The exported information includes physical phenomena, entities, parameters of entities, and mathematical expressions including parameters defined in physical phenomena. For instance, as a starting point, Fig. 6.15 shows a parameter network modeller, which displays the relevant information to analyse the temperature of a sheet of paper through two heating processes. Based on this model, Fig. 6.16 shows the exported information in table format. Figure 6.17 shows the generated model based on the exported information shown in Fig. 6.16. The model becomes executable when the values of all parameters are specified. The values of parameters are specified in SA-CAD (e.g. duration of heating process based on the length of heater and the velocity of paper), although additional information is specified within the context of MATLAB/Simulink (e.g. simulation steps).

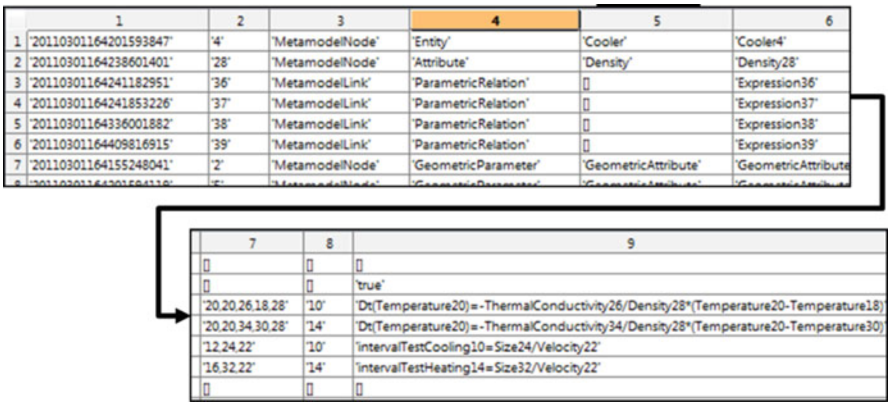
In order for the generation procedure to work, there is a preliminary requirement. The correspondence between building blocks (and their connections) in MATLAB/Simulink on the one hand and the definition of physical phenomena and entities in the knowledge base of SA-CAD on the other hand should be



```
0001: Dt(Temperature20)=-ThermalConductivity26/Density28*(Temperature20    Density28
0002: Dt(Temperature20)=-ThermalConductivity34/Density28*(Temperature20    intervalTestCooling10
0003: intervalTestCooling10=Size24/Velocity22                              intervalTestHeating14
0004: intervalTestHeating14=Size32/Velocity22                              Size24
                                                                           Size32
                                                                           Temperature18

intervalTestHeating14=Size32/Velocity22
```

Fig. 6.15 A parameter network in SA-CAD



	1	2	3	4	5	6	7	8	9
1	'20110301164201593847'	'4'	'MetamodelNode'	'Entity'	'Cooler'	'Cooler4'	[]	[]	[]
2	'20110301164238601401'	'28'	'MetamodelNode'	'Attribute'	'Density'	'Density28'	[]	[]	'true'
3	'20110301164241182951'	'36'	'MetamodelLink'	'ParametricRelation'	[]	'Expression36'	'20,20,26,18,28'	'10'	'Dt(Temperature20)=-ThermalConductivity26/Density28*(Temperature20-Temperature18)'
4	'20110301164241853226'	'37'	'MetamodelLink'	'ParametricRelation'	[]	'Expression37'	'20,20,34,30,28'	'14'	'Dt(Temperature20)=-ThermalConductivity34/Density28*(Temperature20-Temperature30)'
5	'20110301164336001882'	'38'	'MetamodelLink'	'ParametricRelation'	[]	'Expression38'	'12,24,22'	'10'	'intervalTestCooling10=Size24/Velocity22'
6	'20110301164409816915'	'39'	'MetamodelLink'	'ParametricRelation'	[]	'Expression39'	'16,32,22'	'14'	'intervalTestHeating14=Size32/Velocity22'
7	'20110301164155248041'	'2'	'MetamodelNode'	'GeometricParameter'	'GeometricAttribute'	'GeometricAttribute	[]	[]	[]

Fig. 6.16 Exported data in a matrix format

defined in advance. Once the correspondence is established, the architect can simulate the behaviour of a printer using MATLAB/Simulink with different physical decompositions and variation of parameter values.

In the context of this case study, we have successfully generated a simple executable model in MATLAB/Simulink from SA-CAD that concerns the temperature

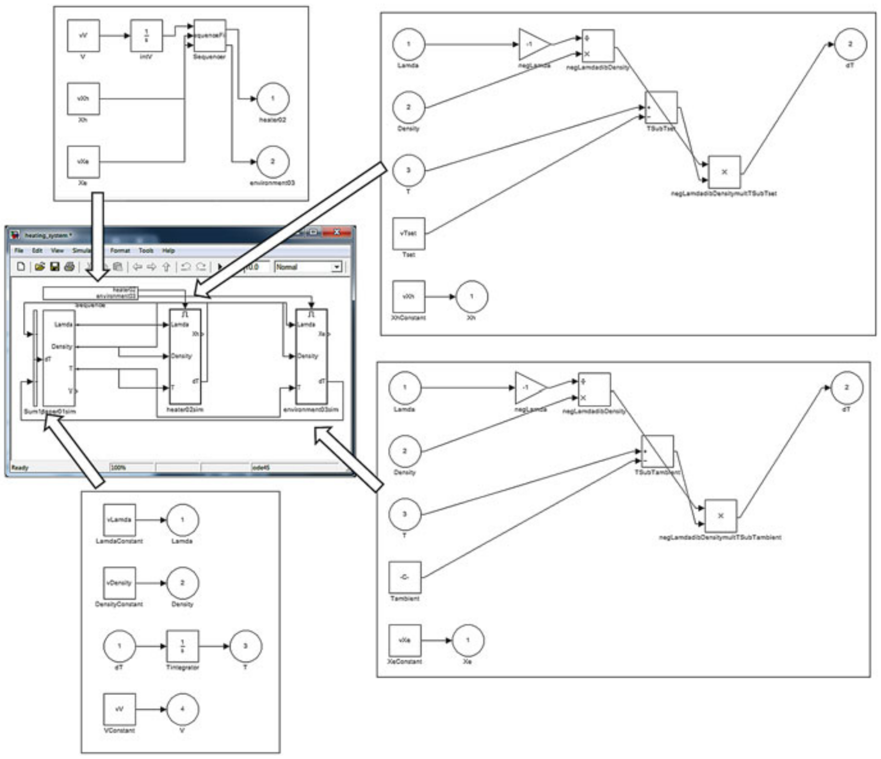


Fig. 6.17 An executable behaviour model in MATLAB/Simulink

of a sheet of paper followed throughout the printing process. In spite of the fact that this was very well feasible, further analysis is necessary to investigate the definition of correspondences that can be applied to the analysis of more complex behaviour. A remaining research question regarding the automated generation procedure is the following one: Is the enrichment of the definition of physical phenomena and entities in the knowledge base of SA-CAD sufficient to cover all structural information and calculation procedures in order to be able to conduct any simulation desired by architects? In the printer case we have seen that this was very well possible.

6.6 Discussion

This section discusses several issues that relate to the proposed systems architecting process and the use of SA-CAD in the context of observed current industrial practices of product development.

Current industrial practice in modelling is observed to be limited to (structural) entities with their quantified properties in order to validate an implicit notion of function. The notion of function is quite stable, but only implicitly present in the heads of designers and engineers. However, we argue that the notion of functions, being the most stable part in the description of a system, is indispensable to design complex systems in a systematic way.

The embodiment of functions into a system, subsystems, and constituent entities results in a connected network of parameters that reflects how interactions take place in the designed system. This is central to verification and validation of the specified functions and requirements that were imposed onto the system.

Architecting is all about decomposition in order to be able to integrate the parts into a system that satisfies the stakeholders' needs. For early phases in product development, we have observed that decomposition over space and over time are two ingredients that deserve special attention: they are important to get grip on the architecting process. The structural and behavioural decomposition of software is a challenge that we left out of the equation in this chapter.

The generation of architectural options has been addressed, based on functional decomposition. This is a promising direction of the research on systems architecting, although this research has just started. One of the challenges is the sheer number of options that have to be dealt with and how generated architectural options can be preliminarily selected in order to be manageable. A further study of the creative process of design may provide insights that relate to a strong reduction of the number of reasonable design options.

Early quantitative evaluation of architectural options is highly prioritised by architects in industry. We believe that the parameter network should play a central role in this. However, the network should be translated into concrete models that can be analysed with well-known, powerful, trusted tools. One example has been given by connecting an SA-CAD model to a MATLAB/Simulink [22] model, which could subsequently be analysed for temporal properties.

The study on systems architecting is related to the research and application of domain-specific languages, because high-level specification and automated generation of either the system artefacts or analysis models are also the central topics of that work (see also Chap. 8).

With SA-CAD, the systems architecting process has been studied from the point-of-view of an architect cooperating with domain experts. The systems architecting support provided by SA-CAD should be extended and aligned with the systems architecting process performed in a large team. For instance, management of mappings among product models used in the systems architecting process (including quantitative models for the verification of architectural options) is one of the crucial functions to be supported. In spite of the extension of SA-CAD in this direction, the concept of SA-CAD, comprised of elements such as the representation of system models, the procedure of systems architecting process, and the computational support used in this procedure, does not change.

6.7 Summary and Conclusions

This chapter has first discussed the roles of architecture in achieving adaptability. It has then presented a method and a tool to support systems architecting of complex multi-disciplinary adaptive systems, which define subsystems and their interfaces through hierarchical decomposition from multiple aspects. The tool and method have been explained with a systems architecting case of a printer, although they are developed such that they can be applied to systems architecting of other products. The case study has shown that the proposed tool supports communication between an architect and domain experts in terms of the function, behaviour, and structure of a printer and management of these notions by means of the parameter network of a printer defined across multiple domains. It has also shown that, following the proposed method, the tool can systematically generate alternative architectural options and quantitative models used for the validation of these options with an analysis tool.

Adaptivity of complex multi-disciplinary systems such as mechatronic products is a system-level quality, which is attained by synthesis of design solutions derived from multiple disciplines. This chapter has illustrated how the synthesis process can be systematically supported with models capturing the symbolic notions and parameter dependencies in the design solutions. The study summarised in the chapter concludes that model-based (model-driven) development of systems architectures is favourable in the design for run-time adaptability in the context of the development of complex multi-disciplinary systems. This chapter has focused on systems architecting to derive "static decompositions". Future work includes how to use the information about a static decomposition to plan and implement "dynamic behaviour" (or action plans) of the system. This is necessary to arrive at truly adaptive architectures.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. 20-sim tooling (2010). http://www.20sim.com/. Accessed May 2012
2. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**, 832–843 (1983)
3. Buede, D.M.: The Engineering Design of Systems: Models and Methods, 2nd edn. Wiley, Hoboken (2009)
4. Dahmus, J.B., Gonzalez-Zugasti, J.P., Otto, K.N.: Modular product architecture. Des. Stud. **22**, 409–424 (2001)
5. D'Amelio, V., Chmarra, M.K., Tomiyama, T.: Early design interference detection based on qualitative physics. Res. Eng. Des. **22**, 223–243 (2011)

6. Dieterle, W.: Mechatronics systems: Automotive applications and modern design methodologies. Ann. Rev. Control **29**, 273–277 (2005)
7. Du, X., Jiao, J., Tseng, M.M.: Architecture of product family: Fundamentals and methodology. Concurr. Eng. **9**, 309–325 (2001)
8. Ferguson, S., Lewis, K., Siddigi, A., de Weck, O.L.: Flexible and reconfigurable systems: Nomenclature and review. In: Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2007), Paper no. DETC2007-35745, pp. 249–263 (2007)
9. Forbus, K.D.: Qualitative process theory. Artif. Intel. **24**, 85–168 (1984)
10. Forsberg, K., Mooz, H.: The relationship of system engineering to the project cycle. In: Proceedings of the Joint conference sponsored by National Council On Systems Engineering (NCOSE) and American Society for Engineering Management (ASEM), Chattanooga, pp. 1–12 (1991)
11. Gu, P., Hashemian, M., Nee, A.Y.C.: Adaptable design. CIRP Ann. Manuf. Technol. **53**, 539–557 (2004)
12. Gu, P., Hashemian, M., Sosale, S., Rivin, E.: An integrated modular design methodology for life-cycle engineering. CIRP Ann. Manuf. Technol. **46**, 71–74 (1997)
13. Henderson, R.M., Clark, K.B.: Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. Adm. Sci. Q. **35**, 9–30 (1990)
14. Holmqvist, T.K.P., Persson, M.L.: Analysis and improvement of product modularization methods: Their ability to deal with complex products. Syst. Eng. **6**, 195–209 (2003)
15. Jiao, J., Simpson, T.W., Siddique, Z.: Product family design and platform-based product development: a state-of-the-art review. J. Intell. Manuf. **18**, 5–29 (2007)
16. Komoto, H., Tomiyama, T.: Computational support for system architecting. In: Proceedings of International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Paper no. DETC2010-28683, pp. 25–34 (2010)
17. Komoto, H., Tomiyama, T.: A system architecting tool for mechatronic systems design. CIRP Ann. Manuf. Technol. **59**, 171–174 (2010)
18. Komoto, H., Tomiyama, T.: Multi-disciplinary system decomposition of complex mechatronics systems. CIRP Ann. Manuf. Technol. **60**, 191–194 (2011)
19. Komoto, H., Tomiyama, T.: A framework for computer-aided conceptual design and its application to system architecting of mechatronics products. Comput. Aided Des. **44**, 931–946 (2012)
20. Maier, M.W., Rechtin, E.: The Art of Systems Architecting, 2nd edn. CRC, Boca Raton (2000)
21. Marshall, R., Leaney, P.G., Botterell, P.: Modular design. Manuf. Eng. **78**, 113–116 (1999)
22. MATLAB/Simulink (2010). http://www.mathworks.com/products/simulink/. Accessed May 2012
23. Modelica Association: Modelica® – a unified object-oriented language for physical systems modeling, Language specification, Version 3.2 (2010)
24. Muller, G.: Systems Architecting: A Business Perspective. CRC, Boca Raton (2011)
25. Olewnik, A., Brauen, T., Ferguson, S., Lewis, K.: A framework for flexible systems and its implementation in multiattribute decision making. ASME J. Mech. Des. **126**, 412–441 (2001)
26. Olewnik, A., Lewis, K.: A decision support framework for flexible system design. J. Eng. Des. **17**, 75–97 (2006)
27. Pahl, G., Beitz, W.: Engineering Design: A Systematic Approach. Springer, Berlin (1988)
28. Palani Rajan, P.K., Van Wie, M., Cambell, M.I., Wood, K.L., Otto, K.N.: An empirical foundation for product flexibility. Des. Stud. **26**, 405–438 (2005)
29. Sage, A.P., Rouse, W.B. (eds.): Handbook of Systems Engineering and Management, 2nd edn. Wiley, Hoboken (2009)
30. Simpson, T.W., Siddique, Z., Jiao, J. (eds.): Product Platform and Product Family Design: Methods and Applications. Springer, New York (2005)
31. Software Engineering Institute: Software architecture overview (2012). URL http://www.sei.cmu.edu/architecture/. Accessed May 2012
32. Stone, R.B., Wood, K.L.: Development of a functional basis for design. J. Mech. Des. **122**, 359–370 (2000)

33. Tomiyama, T., d'Amelio, V., Urbanic, J., ElMaraghy, W.: Complexity of multi-disciplinary design. CIRP Ann. Manuf. Technol. **56**, 185–188 (2007)
34. Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., Tomiyama, T.: Supporting conceptual design based on the function-behavior-state modeler. Artif. Intel. Eng. Des. Anal. Manuf. **10**, 275–288 (1996)
35. Umeda, Y., Kondoh, S., Shimomura, Y., Tomiyama, T.: Development of design methodology for upgradable products based on function-behavior-state modeling. Artif. Intel. Eng. Des. Anal. Manuf. **19**, 161–182 (2005)
36. van de Laar, P., Punter, T. (eds.): Views on Evolvability of Embedded Systems. Springer, Dordrecht (2011)
37. VDI: VDI 2206: Design methodology for mechatronic systems. Beuth Verlag GmbH, Berlin (2004)
38. Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., Tomiyama, T.: Physical concept ontology for the knowledge intensive engineering framework. Adv. Eng. Inform. **18**, 95–113 (2004)

Chapter 7
Model-Driven Design-Space Exploration for Software-Intensive Embedded Systems

Twan Basten, Martijn Hendriks, Nikola Trčka, Lou Somers, Marc Geilen, Yang Yang, Georgeta Igna, Sebastian de Smet, Marc Voorhoeve†, Wil van der Aalst, Henk Corporaal, and Frits Vaandrager

Abstract The complexity of today's embedded systems is increasing rapidly. Ever more functionality is realised in software, for reasons of cost and flexibility. This leads to many implementation alternatives that vary in functionality, performance, hardware, etc. To cope with this complexity, systematic development support during the early phases of design is needed. Model-driven development provides this support. It bridges the gap between ad-hoc back-of-the-envelope or spreadsheet calculations and physical prototypes. Models provide insight in system-level performance characteristics of potential implementation options and are a good means of documentation and communication. They ultimately lead to shorter, more

T. Basten (✉)
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Electronic Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: a.a.basten@tue.nl

M. Hendriks
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: martijn.hendriks@esi.nl

N. Trčka
United Technologies Research Center, 411 Silver Lane, East Hartford, CT 06108, USA

Nikola Trčka was employed at Eindhoven University of Technology when this work was done

L. Somers
Océ-Technologies B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands

Software Engineering and Technology group, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: lou.somers@oce.com

M. Geilen • Y. Yang • H. Corporaal
Electronic Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: m.c.w.geilen@tue.nl; y.yang@tue.nl; h.corporaal@tue.nl

predictable development times and better controlled product quality. This chapter presents the Octopus tool set for model-driven design-space exploration. It supports designers in modelling and analysing design alternatives for embedded software and hardware. It follows the Y-chart paradigm, which advocates a separation between application software functionality, platform implementation choices, and the mapping of software functionality onto the platform. The tool set enables fast and accurate exploration of design alternatives for software-intensive embedded systems.

7.1 Motivation

Industries in the high-tech embedded systems domain (including for example professional printing, lithographic systems, medical imaging, and automotive) are facing the challenge of rapidly increasing complexity of next generations of their systems: Ever more functionality is being added; user expectations regarding quality and reliability increase; an ever tighter integration between the physical processes being controlled and the embedded hardware and software is needed; and technological developments push towards networked, multi-processor and multi-core platforms. The added complexity materialises in the software and hardware embedded at the core of the systems. Important decisions need to be made early in the development trajectory: Which functionality should be realised in software and which in hardware? What is the number and type of processors to be integrated? How should storage (both working memory and disk storage) and transfer of data be organised? Is dedicated hardware development beneficial? How to distribute functionality? How to parallelise software? How can we meet timing, reliability, and robustness requirements? The decisions should take into account the application requirements, cost and time-to-market constraints, as well as aspects like the need to reuse earlier designs or to integrate third-party components.

Industries often adopt some form of model-based design for the software and hardware embedded in their systems. Figure 7.1 illustrates a typical process.

G. Igna • F. Vaandrager
Department of Model-Based System Development, Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
e-mail: g.igna@cs.ru.nl; f.vaandrager@cs.ru.nl

S. de Smet
Océ-Technologies B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands
e-mail: sebastian.desmet@oce.com

M. Voorhoeve • W. van der Aalst
Architecture of Information Systems group, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: w.m.p.v.d.aalst@tue.nl

†Marc Voorhoeve 5 April 1950-7 October 2011

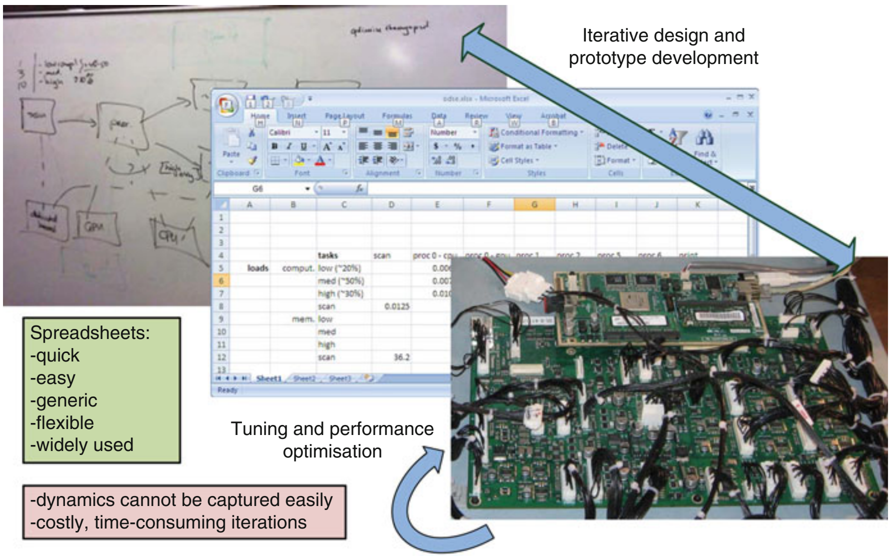

Iterative design and prototype development
Spreadsheets:
-quick
-easy
-generic
-flexible
-widely used
Tuning and performance optimisation
-dynamics cannot be captured easily
-costly, time-consuming iterations

Fig. 7.1 Typical industrial design practice for embedded hardware and software: iterative design and development, intensively using spreadsheets, tuning functionality, and optimising performance in prototypes

Whiteboard and spreadsheet analysis play an important role in early decision making about design alternatives. System decompositions are explored behind a whiteboard. Spreadsheets are then used to capture application workloads and platform characteristics, targeting analysis of average- or worst-case utilisation of platform resources. They provide a quick and easy method to quantitatively explore alternatives from performance and cost perspectives, at a high abstraction level. Promising alternatives are then realised (using various design and coding tools), to validate and fine-tune functionality and performance at the level of an implementation model. Implementation models typically realise important parts of the functionality, they integrate real code, and may run on prototype hardware. The entire process may be iterated several times before arriving at the final result.

Design iterations through prototypes are time-consuming and costly. Only a few design alternatives can be explored in detail. The number of design alternatives is however extremely large. The challenge is therefore to effectively handle these many possibilities, without loosing interesting options, and avoiding design iterations and extensive tuning and re-engineering at the implementation level. Spreadsheet analysis is suitable for a coarse pruning of options. However, it is not well suited to capture system dynamics due to for example pipelined, parallel processing, data-dependent workload variations, scheduling and arbitration on shared resources, variations in data granularity, etc. (see Fig. 7.2).

Understanding and analysing the pipelined, parallel processing of dynamic streams of data is challenging. The relation between design parameters (such as the number and type of processing units, memory size and organisation, intercon-

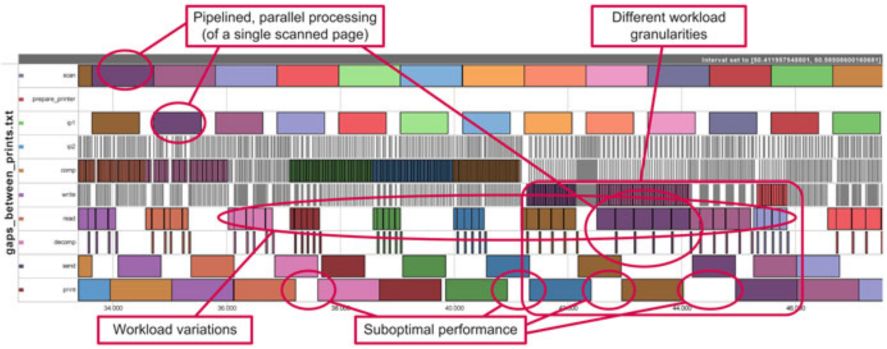


Fig. 7.2 A Gantt chart showing the execution of a print pipeline. Dynamics in the processing pipeline cause hick-ups in print performance due to under-dimensioning of the embedded execution platform (figure from [7])

nect, scheduling and arbitration policies) and metrics of interest (timing, resource utilisation, energy usage, cost, etc.) is often difficult to establish. An important challenge in embedded-system design is therefore to find the right abstractions to support accurate and extensive design-space exploration (DSE).

This chapter presents an approach to model-driven DSE and a supporting tool set, the Octopus tool set. The approach targets an abstraction level that captures the important dynamics while omitting the detailed functional and operational behaviour. The abstractions bridge the gap between spreadsheet analysis and implementation models and prototypes. The approach is designed to specifically cope with the challenges of DSE.

An important characteristic of DSE is that many different questions may need to be answered, related to system architecture and dimensioning, resource cost and performance of various design alternatives, identification of performance bottlenecks, sensitivity to workload variations or spec changes, energy efficiency, etc. Different models may be required to address these questions. Models should be intuitive to develop for engineers, potentially from different disciplines (hardware, software, control), and they should be consistent with each other. Multiple tools may be needed to support the modelling and analysis.

Given these characteristics, our approach to address the challenges of model-driven DSE is based on two important principles: (1) separation of concerns and (2) reuse and integration of existing techniques and tools. The modelling follows the Y-chart paradigm of [6, 39] (see Fig. 7.3) that separates the concerns of modelling the application functionality, the embedded platform, and the mapping of application functionality onto the platform. This separation allows to explore variations in some of these aspects, for example the platform configuration or the resource arbitration, while fixing other aspects, such as the parallelised task structure of the application. It also facilitates reuse of aspect models over different designs. The tool set architecture separates the modelling of design alternatives, their analysis,

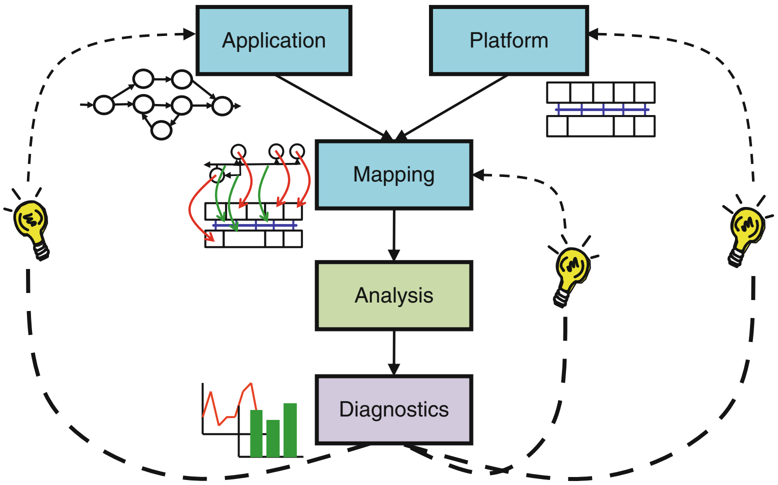


Fig. 7.3 The Y-chart paradigm for design-space exploration separates the modelling of application functionality, platform functionality, and mapping of application functionality onto the platform; after analysis and diagnostics, any of these aspects may be changed to explore alternatives (either automatically or interactively by a designer) (Y-chart: [6, 39]; figure from [8])

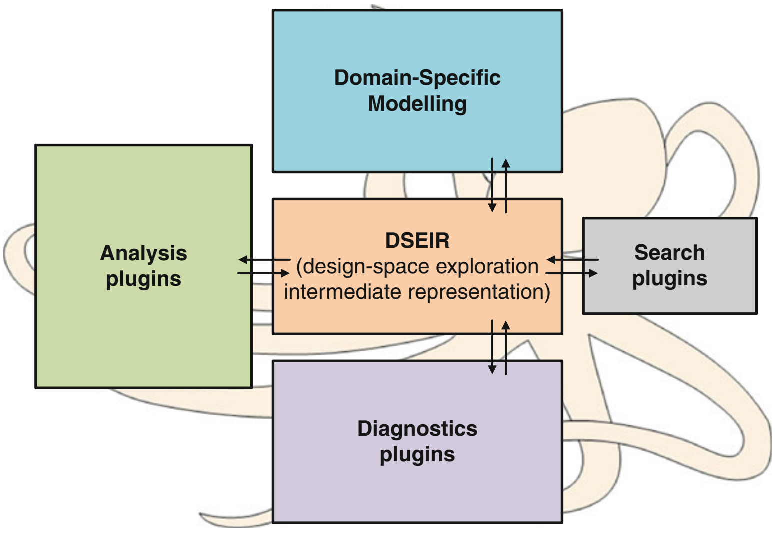

Domain-Specific Modelling
Analysis plugins
DSEIR (design-space exploration intermediate representation)
Search plugins
Diagnostics plugins

Fig. 7.4 Top view of the Octopus tool set (figure from [7])

the interpretation and diagnostics of analysis results, and the exploration of the space of alternatives (see Fig. 7.4). This separation is obtained by introducing an intermediate representation, the DSE Intermediate Representation (DSEIR), and automatic model transformations to and from this representation. This setup allows the use of a flexible combination of models and tools. It supports domain-specific modelling in combination with generic analysis tools. Multiple analyses can be applied on the same model, guaranteeing model consistency among these analyses; different analysis types and analyses based on multiple models can be integrated in a single search of the design space. Results can be interpreted in a unified diagnostics framework.

Chapter overview. Section 7.2 provides an overview of the challenges we have encountered in modelling and analysis support for taking design decisions during early development. The experience draws upon our work in the professional printing domain, but the challenges are valid for a wide range of high-tech embedded systems. Section 7.3 explains the model-driven DSE approach we propose to handle these challenges. This section also surveys related work. To illustrate the possibilities for domain-specific modelling, Sect. 7.4 presents DPML, the Data Path Modelling Language, which is a domain-specific modelling language for the printing domain. A DSE case study from the professional printing domain is introduced to illustrate DPML. Section 7.5 introduces the intermediate representation DSEIR, which is at the core of the Octopus tool set. Section 7.6 presents model transformations to a number of analysis tools. Section 7.7 illustrates the support integrated in Octopus for interpretation and diagnostics of analysis results. Section 7.8 briefly discusses some implementation choices underlying the tool set. Section 7.9 presents the results we obtained in several industrial case studies. Section 7.10 concludes.

Bibliographical notes. An extended abstract of this chapter appeared as [7]. The Octopus tool set was first described in [8]. Our philosophy behind model-driven DSE was originally presented in [69]. Sections 7.2 and 7.3 are based on [69]. Section 7.4 describing DPML is based on [65], which provides a more elaborate description of DPML. DSEIR, summarised in Sect. 7.5, is described in more detail in [69].

7.2 Challenges in Early Design

Making the right decisions early in the design process of a complex software-intensive embedded system is a difficult task. In this section, we discuss the challenges we faced while conducting several case studies at Océ-Technologies, involving the design of digital data paths in professional printers. These challenges are characteristic for other application domains as well.

Multi-functional printing systems perform a variety of image processing functions on digital documents that support the standard scanning, copying, and printing use cases. The digital data path encompasses the complete trajectory of the image data from source (for example the scanner or the network) to target (the imaging unit or the network).

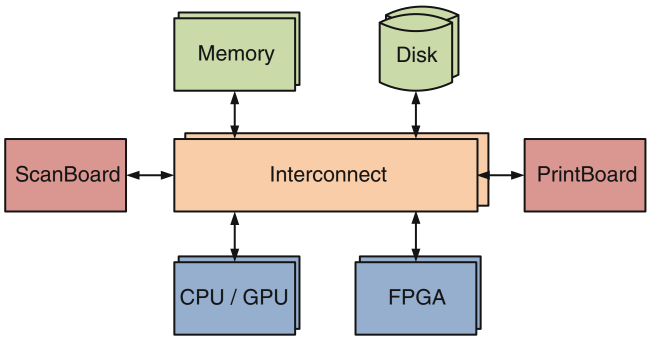


Fig. 7.5 A template for a typical printer data path architecture (figure from [69])

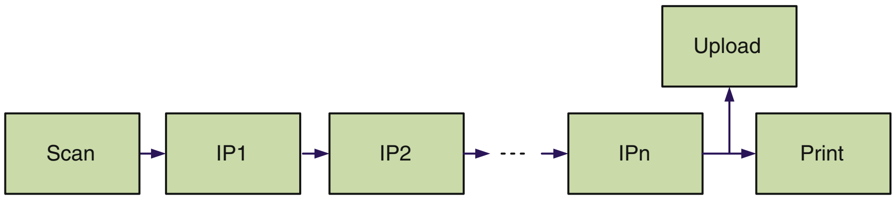


Fig. 7.6 An example printer use case: copying

Data path platform template. Figure 7.5 shows a template of a typical embedded platform architecture for the digital data path of a professional printer. Several special-purpose boards are used to perform dedicated tasks, typically directly related to the actual scanning and printing. For computation, the data path platform may provide both general-purpose processors (CPUs, GPUs) and special-purpose FPGA-based boards. RAM memory and hard disks are used for temporary and persistent storage. The components are connected by interconnect infrastructure (e.g. PCI, USB). The architecture template shows the components needed for the digital image processing, leaving out user controls, network interfaces, etc. Note that the template is in fact generic for almost any modern software-intensive embedded system.

Printer use cases. Each printer needs to support dozens of use cases. The standard ones are scanning, printing, copying, scan-to-email, and print-from-disk. Each use case typically involves several image processing steps such as rendering, zooming, rotating, compressing, half-toning, etc.; these steps may need several components in the platform, and different choices for implementing these steps may be possible. Moreover, print and scan use cases can be mixed. They can also be instantiated for documents with different paper sizes, numbers and types of pages, etc. It is clear that this results in an explosion of possibilities.

As an illustration, we sketch the copying use case in some more detail; see Fig. 7.6. After scanning, each page is first processed by the ScanBoard (that implements the Scan task) and then further processed in several image processing steps (IP1…IPn); the resulting image is then printed by the PrintBoard (executing the Print task). The image processing steps need to be done on some computation resource. Intermediate results are stored in memory and/or on disk. The latter is done for example to allow more than one copy of the document to be printed or to cope with errors. Uploading a processed image to disk can be done in parallel with any further processing.

Questions in data path design. The ScanBoard and PrintBoard are typically the first components to be selected for a specific printer. They determine the maximum possible scan and print speeds in pages per minute. The rest of the data path should be designed in such a way that these scan and print speeds are realised at minimum cost.

Typically, a number of questions need to be answered early in the development trajectory. Which types of processing units should be used? How many? What clock speeds are needed? What amount of memory is needed? Which, and how many, buses are required? How should image processing steps be mapped onto the resources? Other questions relate to scheduling, resource allocation, and arbitration. What should be the scheduling and arbitration policies on shared resources? What are the appropriate task priorities? Can we apply page caching to improve the performance? How to allocate memory in RAM? How to share memory? How to minimise buffering between tasks? How to mask workload variations? Is the memory allocation policy free of deadlocks?

The data path design should take into account all basic use cases, as well as combinations such as simultaneous printing and scanning. It should also take into account different job types (text, images), paper sizes, etc. The design should be such that no bottlenecks are created for the most important use cases (normal printing, scanning, and copying, on the default paper size for normal jobs). Performance can be traded off for costs for use cases that occur less frequently though. DSE should also provide insight in these trade-offs. Furthermore, printing products typically evolve over time. This raises questions such as what is the impact of a new scan- or print board with higher specs on an existing data path design. It is clear that DSE complexity is large and that quantifying all the mentioned aspects early in the design process is challenging.

Modelling dynamics. Modelling the above use cases for DSE is possible with a spreadsheet at a high level of abstraction as a first-order approximation. Spreadsheet estimates, however, may lead to over- or under-dimensioning of the ultimate design, which is costly to repair in later design stages. The cause for over- or under-dimensioning is the difficulty to capture various dynamic aspects of software-intensive embedded systems in a spreadsheet. First, there are various sources of variability. The complexity of a page to be printed, the size of a compressed page, and the execution time on a general-purpose processor are all stochastic and rarely exactly predictable. Second, the flow of information is often iterative or conditional. An example of a conditional flow is a smart-storage heuristic that takes a page from disk only if it is not still in memory. Third, pipelined and parallel steps in a job and simultaneously active print and scan jobs may dynamically interact on shared resources such as memory, buses, and shared processors. Finally, scheduling and arbitration policies are often crucial for performance, but result in dynamic behaviour that is hard if not impossible to model in a spreadsheet-type model.

Mixed abstraction levels. Although many image processing steps work on pixels or lines, parts of most use cases can be accurately modelled at the page level. The throughput of the data path in images per minute is also the most important metric of interest. A mixture of abstraction levels may be needed to achieve the required accuracy while maintaining analysis efficiency. FPGAs for example come with limited memory sizes. Only a limited number of lines of a page fit in FPGA memory. The page level thus becomes too coarse and a finer granularity of modelling is needed. Modelling complete use cases at the line or pixel level would make most analyses intractable though; appropriate transitions between abstraction levels are needed.

Variety in analysis questions. The typical analysis questions in the list of DSE questions above may, in theory, all potentially be answered by a generic modelling and analysis tool; it is clear, however, that the various types of DSE questions may be of a very different nature. Deadlock and schedulability checks, for example, are best done using a model checker. Performance analysis would typically be done with analytic models, like spreadsheets, for a coarse evaluation and simulation for a more refined analysis that takes into account the dynamics in the system. Low-level FPGA buffer optimisation can be done with fast, yet restrictive, dataflow analysis. This variety in analysis questions suggests the use of different tools. This does require the development of multiple models though, leading to extra modelling effort and a risk of model inconsistencies and interpretation difficulties. Ideally, one master model would form a basis for analyses performed with different techniques and tools.

Model parametrisation. There is a need to support a high degree of parametrisation of models: Multiple use cases need to be captured, each of them with many variations; models are needed for various DSE questions; design decisions may need to be reconsidered to cope with late design changes; and to speed up development, there is a clear wish to reuse models across variations of the same product, both to allow product customisation and to support the development of product families. The desired parametrisation goes beyond simple parameters capturing for example workloads, task execution times, and memory requirements; they should also cover for example the flow of use case processing, communication mechanisms, platform resources, and scheduling and arbitration policies.

Customisation for the printer domain. The basic principles of printer platforms and printer use cases are not rapidly changing. Having the models written in a printer-specific language, and maintaining a library of those models, would drastically decrease the modelling effort for new printers, reduce modelling errors, and improve communication and documentation of design choices. The design of a domain-specific language for the printer domain is challenging. On the one hand, we want a simple language, which only contains constructs that are needed to describe the current designs. On the other hand, it should also be possible to use (simple extensions of) the language to describe the designs of tomorrow.

7.3 Model-Driven Design-Space Exploration

The previous section clarified the challenges in early DSE. In this section, we first identify the main benefits of a model-driven approach to tackling these challenges. We then set out the key ingredients of our approach to model-driven DSE. Along the way, we survey methods, languages, and tools that fit in such an approach. The following sections then elaborate on the Octopus tool set that is being developed to support the model-driven DSE approach and that integrates several of the surveyed methods, languages, and tools.

The benefits of model-driven DSE. The ultimate objective of model-driven DSE is to reduce development time (and thereby time-to-market), while maintaining or improving product quality. This is achieved by appropriate modelling and analysis during early development. Models should capture the essential system dynamics without including unnecessary details. Only then, effective exploration of design alternatives is feasible. Figure 7.7 visualises the model-driven DSE approach and summarises the targeted benefits. With appropriate modelling and tool support, (1) insight in system dynamics and design alternatives improves, (2) models are reusable within the product development trajectory and across developments, (3) it becomes feasible to apply different analyses, (4) different models may be used while safeguarding their consistency, and (5) documentation improves by using the models themselves as the design documentation. In combination, these benefits lead to (6) the intended reduction in development time.

Separation of concerns. To address the challenges outlined in the previous section and to realise the above-mentioned benefits, we propose a rigourous separation of concerns.

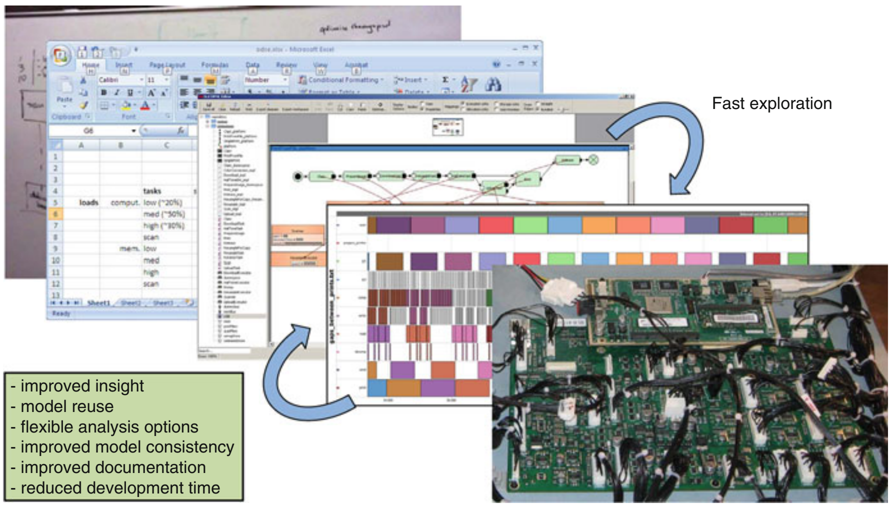

Fast exploration
- improved insight
- model reuse
- flexible analysis options
- improved model consistency
- improved documentation
- reduced development time

Fig. 7.7 Model-driven design-space exploration

First, the tool set organisation should separate the modelling, analysis, diagnostics, and search activities, as already illustrated in Fig. 7.4. Modules for each of these activities are decoupled through an intermediate representation, DSEIR (DSE Intermediate Representation; see Sect. 7.5). Such an organisation realises the required flexibility in modelling and analysis needs. The use of an intermediate representation allows reuse of analysis techniques and tools across different models and in combination with different modelling environments. Model consistency is ensured by appropriate model transformations to and from the intermediate representation. The challenge is to develop an intermediate representation that is sufficiently rich to support DSE but not so complex that it prohibits model transformations to various analysis techniques and tools. These model transformations should preserve precisely defined properties, so that analysis results from different tools can be combined and results can be interpreted in the original model.

Second, modelling should follow the Y-chart philosophy [6,39]. This philosophy is based on the observation that DSE typically involves the co-development of an application, a platform, and the mapping of the application onto the platform (as already illustrated in Fig. 7.3). Diagnostic information is then used to, automatically or manually, improve application, platform, and/or mapping. This separation of application, platform, and mapping is important to allow independent evaluation of various alternatives of one of these system aspects while keeping the others fixed the others. Often, for example, various platform and mapping options are investigated for a fixed set of applications. DSEIR separates the application, platform, and mapping modelling. In combination with the tool set organisation illustrated in Fig. 7.4, DSEIR thus supports the Y-chart philosophy.

Application-centric domain-specific modelling. A key challenge in modelling for DSE is to fully take into account the relevant system dynamics such as realisable concurrency, variations in application behaviour, and resource behaviour and sharing. Given the complexity of the DSE process, a modelling abstraction level is needed that abstracts from implementation details but is more refined than typical spreadsheet-type analysis. Modelling should be simple and allow efficient and accurate analysis.

Another important aspect in modelling for DSE is that the abstractions need to appeal to the designer and adhere to his or her intuition. Domain-specific abstractions and customisation should therefore be supported. Modelling should furthermore be application-centric. The application functionality and the quality (performance, energy efficiency, reliability) with which it is provided is what is visible to users and customers. Application functionality should therefore be leading, and the models should capture all behaviour variation and all concurrency explicitly. We propose to model platforms as sets of resources that have no behaviour of their own; their purpose is only to (further) restrict application behaviour and to introduce proper timing. This leads to simple, predictable, and tractable models. Scheduling and mapping of applications onto resources can then be unified into

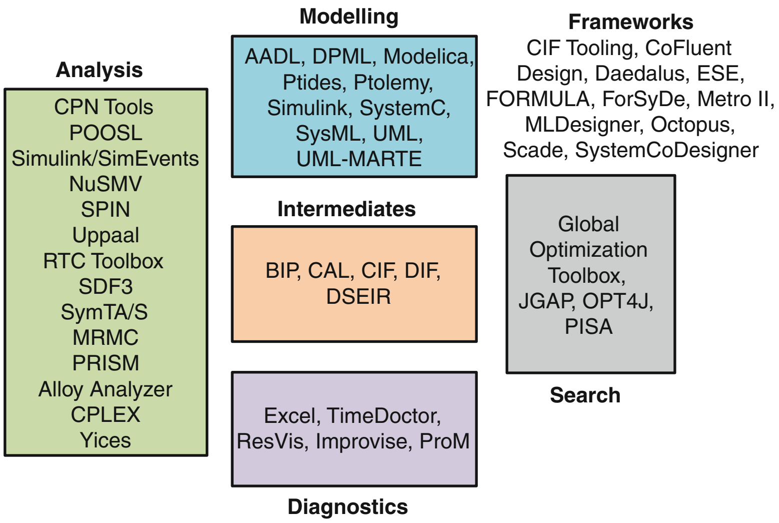

Modelling
AADL, DPML, Modelica, Ptides, Ptolemy, Simulink, SystemC, SysML, UML, UML-MARTE
Frameworks
CIF Tooling, CoFluent Design, Daedalus, ESE, FORMULA, ForSyDe, Metro II, MLDesigner, Octopus, Scade, SystemCoDesigner
Analysis
CPN Tools
POOSL
Simulink/SimEvents
NuSMV
SPIN
Uppaal
RTC Toolbox
SDF3
SymTA/S
MRMC
PRISM
Alloy Analyzer
CPLEX
Yices
Intermediates
BIP, CAL, CIF, DIF, DSEIR
Global Optimization Toolbox, JGAP, OPT4J, PISA
Search
Excel, TimeDoctor, ResVis, Improvise, ProM
Diagnostics

Fig. 7.8 Methods, languages, and tools that fit in the *top*-level architectural view of Fig. 7.4. Every entry is only mentioned where it fits best in the architectural view (even though it may fit in other places as well) (figure adapted from [69])

the concept of prioritised dynamic binding. If needed, complex resource behaviour (work division, run-time reconfiguration, etc.) can be modelled through (automatic) translations into application behaviour.

A variety of modelling environments and approaches in use today, either in industry or in academia, can support the envisioned modelling style. We mention some of them, without claiming to be complete: AADL [1], DPML [65], Modelica [47], Ptides [19], Ptolemy [21], MATLAB/Simulink [45], SysML [64], SystemC [51], UML [72], and UML-MARTE [73]. The mentioned environments often target different application domains and/or different system aspects. Figure 7.8 positions these modelling approaches in the architectural framework of the Octopus tool set. DPML, the Data Path Modelling Language, is discussed in more detail in Sect. 7.4.

Analysis, search, diagnostics. The previous section illustrated the variety of design questions and challenges that are typically encountered early during development. No single tool or analysis method is fit to address all these questions. We foresee the combined use of different analysis tools in one DSE process. A wide variety of, mostly academic, but also some commercial, tools is available that can be used to support DSE.

For quick exploration and performance optimisation, discrete-event simulators such as CPN Tools [35], POOSL [66], and Simulink/SimEvents [44,45] are suitable. Model checkers such as NuSMV [50], SPIN [25], and Uppaal [10] can be used for functional verification, protocol checking, and schedule and timing optimisation. Model checkers may not be able to cope with the full complexity of modern embedded systems, but they may play an important role in verifying and optimising critical parts of the system. Yet other tools, such as the RTC Toolbox [57], SDF3 [62], and SymTA/S [63], are suited for timing analysis of data-intensive system parts, such as image processing chains.

Questions regarding performance, reliability, and schedulability under soft deadlines can be answered by increasingly popular probabilistic model checking techniques, using tools like PRISM [54] and MRMC [37]. These techniques enhance the expressivity of regular model checking, allowing for more realistic modelling of aspects such as arrival rates and failure times. Scalability of these techniques to realistic analysis problems remains a challenge though.

In recent years, also constraint programming and SAT/SMT solvers have gained popularity. With the rise of more powerful computers and improvements in the techniques themselves, tools like CPLEX [28], Alloy Analyzer [32], and Yices [80] are increasingly often used to find optimal or feasible solutions for system aspects such as resource bindings or schedules.

The Octopus tool set has links to three analysis tools, namely CPN Tools, Uppaal, and SDF3. The model transformations that realise these links and the intended use of these tools in the Octopus context are discussed in Sect. 7.6.

Besides support for evaluation of metrics for design alternatives or for the optimisation of parts of the system, we also need support to explore the large space of design alternatives. The MathWorks Global Optimization Toolbox [43] supports a wide variety of customisable search algorithms. The JGAP library [34] is a Java library for developing genetic search algorithms. OPT4J [42] and PISA [11] are customisable genetic search frameworks that support DSE.

Most of the tools mentioned above already give good diagnostic reports, which in many cases can be successfully converted and interpreted in the original domain. Microsoft Excel is also a useful tool in this context. Visualisation of Gantt charts (using tools such as TimeDoctor [68] or ResVis [59], from which the screenshot of Fig. 7.2 is taken) helps understanding the dynamic behaviour of design alternatives. Sophisticated mining and visualisation is possible with Improvise [31] or ProM [55]. Section 7.7 presents the diagnostic support as it is developed in the Octopus context, which includes Gantt chart visualisation through ResVis.

Intermediate representation: flexibility, consistency, customisation. It cannot be expected that designers master the wide variety of modelling languages and tools mentioned so far, and apply them in combination in DSE. To successfully deal with integration, customisation, and adaptation of models, as well as to facilitate the reuse of models across tools and to ensure consistency between models, we foresee the need for an intermediate representation to connect different languages and tools in a DSE process. Such an intermediate representation must in the first place be able to model the three main ingredients of the Y-chart (application, platform, mapping) in an explicit form. It should not have too many specific constructs to facilitate translation from different domain-specific modelling languages and to different target analysis tools, yet it must be powerful and expressive enough to accommodate developers. A good balance between modelling expressiveness and language complexity is needed. Besides the Y-chart parts, the intermediate representation must provide generic means to specify sets of design alternatives, quantitative and qualitative properties, experimental setups, diagnostic information, etc., i.e. all ingredients of a DSE process. The intermediate representation should have a formal semantic basis, to avoid interpretation problems and ambiguity between different models and analysis results. The intermediate representation does not necessarily need execution support, because execution can be done through back-end analysis tools. Intermediate representations and languages like BIP [9], CAL [20], CIF [74], DIF [27] and the intermediate representation DSEIR (see Sect. 7.5) underlying the Octopus tool set are examples of languages that can be adapted to fully support model-driven DSE as sketched in this section.

A DSE tool set. To realise the goals set out, it is important to provide a flexible tool set implementation. We propose a service-based implementation of the tool set architecture of Fig. 7.4. Modules should communicate with other modules through clean service interfaces. Domain-specific modelling tools with import/export facilities to DSEIR are in the modelling module. The analysis module provides analysis services such as performance evaluation, formal verification, mapping optimisation, schedulability analysis, etc. The diagnostics module provides ways to visualise analysis results and gives high-level interpretations of system dynamics in a way intuitive to system designers. The search module contains support for search techniques to be used during DSE. Information flows between the modules go through the DSEIR kernel module that implements the intermediate representation.

There are several frameworks and tool sets that support DSE, or aspects of it, for various application domains. Examples are CoFluent Design [15], Daedalus [49], ESE [75], FORMULA [33], ForSyDe [58], METRO II [17], MLDesigner [46], SCADE [22], and SystemCoDesigner [38]. The large number of available languages, tools, and frameworks are a clear indication of the potential of high-level modelling, analysis, and DSE. The Octopus tool set, described in more detail in the remainder of this chapter, is closest to the views outlined in this section. Octopus explicitly aims to leverage the combined strengths of existing tools and methods in DSE. Its service-based implementation is discussed in Sect. 7.8.

7.4 DPML: Data Path Modelling Language

The entry point of our tool chain is the modelling module (see Fig. 7.4). Models can be developed in a domain-specific language (DSL), that functions as a front end for the tool chain. For modelling printer data paths, we have designed DPML (Data Path Modelling Language). This section first introduces a typical DSE case study as a running example. It then presents the design goals for DPML, followed by an overview of DPML and a presentation of the main DPML concepts along the lines of the Y-chart separation of concerns (see Fig. 7.3).

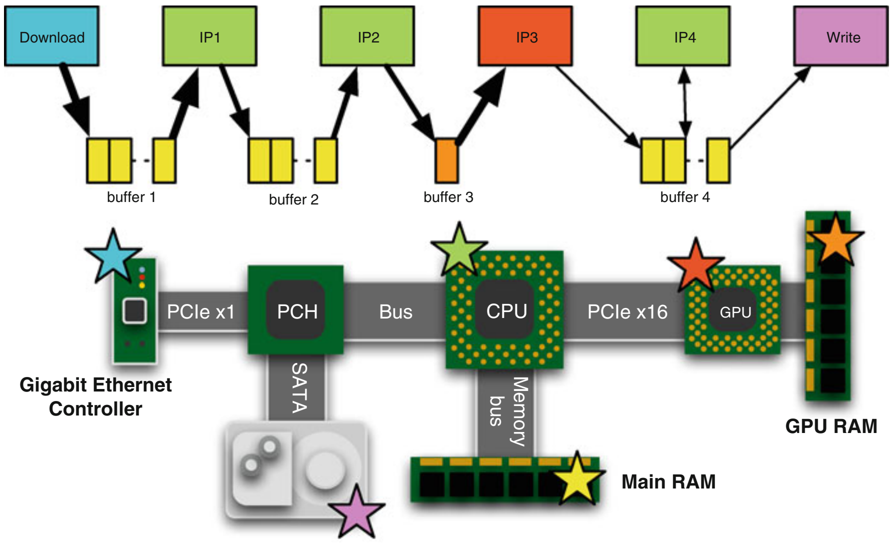


Fig. 7.9 High-level architecture of a part of the scan path in a high-end colour copier

7.4.1 *Running Example: High-End Colour Copier*

The scan path of a copier is the part of the data path that receives data from the scanner hardware, processes them, and stores them for later use (e.g. sending them to the printer hardware or to an e-mail account). Figure 7.9 shows the high-level architecture of a part of the scan path in a high-end colour copier. Its structure follows the Y-chart approach. The application consists of six tasks: a step that downloads image data from the scanner hardware, four image processing steps IP1, . . ., IP4, and a step that writes the processed image data to disk. The tasks pass image data through various statically allocated buffers (i.e. the buffer slot size is constant and both the size and the number of slots are determined at design time). Note that task IP4 reads from and writes to the same buffer slot. The platform consists of a general-purpose board with CPUs/GPUs and potentially several dedicated boards, for example for the interface with the print engine (not shown in the figure). The mapping of the tasks and the buffers to the platform is depicted by the colour-star combination. Steps IP1, IP2, and IP4 all use the CPU, whereas the other steps each have their own computational resource.

There are several important aspects that complicate the modelling and analysis:

- Various steps use different data granularities (indicated in the figure by the different widths of the arrows to and from the buffers). The download step writes the data of a complete image in a buffer slot of buffer 1. Step IP1 processes these data and outputs the result in ten so-called *bands*. Step IP2 processes each band and appends it to the single slot of buffer 3. As soon as the data of a complete

image are present in buffer 3, step IP3 processes them and writes the result in finer grained bands to buffer 4. Steps IP4 and *Write to disk* process these bands individually.
- Buffers 1, 2, and 4 can have multiple slots which allows pipelining of the processing steps of consecutive images.
- The scheduling on the CPU is priority-based preemptive.
- The execution times of the steps are not known exactly, and often heavily depend on the input image and on the result of the algorithms in the steps that change the data size (e.g. compression).

The main performance indicator of the data path is *throughput*. It is important that the data path is not the limiting factor of the machine. For cost reasons, the scan hardware should be fully utilised. Another, albeit less important, performance indicator is the total amount of main memory that buffers 1, 2, and 4 consume. Since memory is a scarce resource, the buffers should not be over-dimensioned. The DSE question is therefore as follows:

> Minimise the amount of memory allocated to buffers 1, 2, and 4 while retaining a minimum given throughput.

7.4.2 The DPML Design Goals

DPML is intended to support modelling for DSE of printer data paths. When designing DPML, four goals were kept in mind:

- First, DPML must be particularly suited to analyse the *speed* (or throughput) of data path designs. This means that all information necessary for obtaining the speed of a data path must be present, but behavioural issues that do not influence speed may be abstracted away. The data sizes of images and image parts being manipulated and transferred play a dominant role.
- Furthermore, DPML has to be *expressive* and *flexible*. This means that it must be possible to express a wide variety of models, with different behavioural and structural properties. This is important, because we cannot always foresee what kinds of designs engineers may want to analyse in the future, or what other purposes (than analysing speed) may be found for DPML models. Therefore, it must be easy to model many things in DPML, and it must also be easy to change DPML to add more features. This requirement is essential if the tool chain is to be a sustainable solution.
- DPML has to *closely match the problem domain*. This means that all elements commonly found in printer data paths must be well supported and easy to model. The most visible example of this is the concept of pages flowing through the steps of the application. Without this, even simple designs would require considerable modelling effort and thus a steeper learning curve for people using the tools for the first time. This may in turn impact the adoption of the tool set as a means to improve the data path design process.

Important are also the features that DPML does *not* have; features that would make it a more generic specification language, such as support for caches, the ability to fully specify the behaviour of steps, the possibility to specify real-time deadlines, or the ability to fine-tune a task scheduler. Leaving out such features makes DPML a simple language, in which models of commonly occurring data path designs are not significantly more complex than what an engineer would draw in an informal sketch of the same design.

- Finally, DPML has to support *modular* designs. This way, parts, or elements that are used in multiple designs can be reused, thus saving modelling time. Additionally, a modular setup allows engineers to share knowledge obtained during design or engineering (such as the actual speed of a hardware component, or the rationale behind a design decision) with engineers in other projects who may use some of the same parts.

7.4.3 DPML Overview

DPML is a combined visual and textual language. The visual parts outline the coarse structure of a data path design, such as the series of processing steps that a data path may be required to perform, and the component layout of a hardware platform. These are represented visually so that, even in large models, it is easy to get a good overview. The textual parts are used to express all details of every element in a data path design, such as the behaviour of a step or the capacity of a memory. These details are expressed with a custom, text-based language so that it is easy to add new constructs and features.

Structurally, a complete DPML model consists of the three distinct components of the Y-chart paradigm:

- An **application**, which functionally describes a series of steps that a data path may be required to perform.
- A **platform**, which describes the various hardware components that a data path consists of and how they are laid out.
- A **mapping** between these two, which describes which hardware components are used by which steps, and how.

DPML models can be edited in a custom editor that is based on the open source Qt library [56]. A single DPML model is contained in multiple small files, each of which describes a reusable element. A textual DPML element is stored as a plain text file and visual DPML elements are stored in a custom XML format. To facilitate the analysis and further conversion of DPML models, e.g. to the Octopus intermediate representation DSEIR, these files are converted to a simpler data format, DPML Compact. DPML Compact models can be simulated using a native simulator. The advantage of having this compact format is that changes in and additions to the language do not affect the simulator and the model transformations as long as DPML Compact remains unchanged.

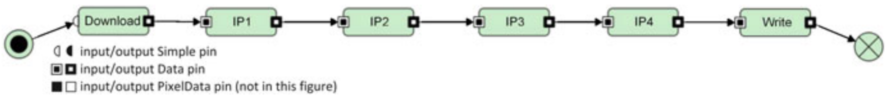


Fig. 7.10 A DPML application

7.4.4 *DPML: The Application View*

Figure 7.10 displays the application component of our running example described in DPML. The model captures the pipelined processing of scan jobs consisting of any number of scanned pages. Every rounded rectangle in Fig. 7.10 is a **step**, which is a single image processing operation that the data path must perform on a single image or part of an image. Each step has two or more **pins**, the squares or circles on the sides.

Pins can have three possible types: `Simple`, `Data`, and `PixelData`. Visually, a `Simple` pin is a semicircle, a `Data` pin is a square, and a `PixelData` pin is a square with a small square in the middle. Every `PixelData` pin also is a `Data` pin, and every `Data` pin is also a `Simple` pin. When a step has an output `Data` pin, this means that this step *produces data* at that pin. If it is a `PixelData` pin, this additionally implies that the data produced represent a bitmap image. Similarly, when a step has an input `Data` pin, it means that this *consumes data* at that pin.

Arcs between steps indicate data dependencies. They further determine the execution order of the application. By default, a step can only start when for all its input pins, the preceding step has completed. This explains the need for `Simple` input pins; no data are consumed or produced on those pins, but they can be used to fix the execution order between steps.

Because a printer data path is all about image processing, we assume that we can model the duration of a single image processing step as a function of the size of the image data, the speed and availability of all resources used, and predefined information about the pages that are to be processed. Most notably, we assume that it does not depend on the *precise content* of the image. This assumption is important, because it means that instead of formally specifying all aspects of a step behaviour, it is sufficient to just specify the *data sizes* that it produces.

In DPML, a **task** is used to describe a single operation that we may want a data path to perform. Each step in an application is in fact an *instantiation* of such a task: A step behaves exactly as dictated by the task, and each step corresponds to exactly one task. It is, however, possible for multiple steps to belong to the same task. Multiple compression step instances of the same task may for example occur in a single image processing pipeline. Because a step cannot exist without a task, the set of available tasks defines the set of operations we can use in an application. The relationship between tasks and steps is therefore somewhat comparable to the relationship between classes and objects in object-oriented programming languages.

Tasks are stored as small text files with content as shown in Fig. 7.11. A task definition has three parts: a header, a set of pin declarations, and a set of

```
task Resample
{
  inpin  in:  PixelData;
  outpin out: PixelData;

  out.width    = in.width  * page.zoomFactorX;
  out.length   = in.length * page.zoomFactorY;
  out.bitdepth = in.bitdepth;
  precondition = true; //optional and redundant
}
```

Fig. 7.11 An example of a task in DPML (figure from [65])

properties. The header defines the name of the task, in this case “Resample”. The pin declarations define the number of input and output pins, their names, and their data types. The Resample task takes a raster image and produces a raster image, so there is one input pin and one output pin, both of type `PixelData`. The properties constitute the functional description of the behaviour of a task. Because we assume that the actual content of the image produced does not matter, we only need to determine the size of the output image (so in fact we are only describing a very small part of the required behaviour, focusing on resource usage and performance aspects).

In the example of the Resample task, the width and length of the output image depend on the width and length of the input image as well as a user setting, the zoom factor. Because it is possible for a single scan job to require some pages to be zoomed and some pages not, the zoom setting is looked up as a property of the current page.

Tasks can specify more properties than just the sizes of its output images, such as a boolean condition that must be true for a task to be able to start. All of these other properties are optional.

7.4.5 DPML: The Platform View

A platform defines the hardware necessary to perform the steps in a data path. Such hardware typically includes processors, memories, hard drives, caches, cables and buses, and of course the actual printing and scanning components. In DPML, these components belong to a set of three resource types:

- A **memory** is something that can store and retrieve data, so it includes hardware such as RAM chips and hard drives. A memory has a particular capacity, which is its only limiting factor.
- A **bus** is something that can transfer data. A bus typically has a maximum bandwidth (the maximum number of bytes it can transfer per second).
- An **executor** is something that can execute a step and has some processing speed. Executors are subdivided into processors, scanners, and printers for clarity, but from a semantics point of view there is no difference between a processor, a scanner, and a printer in DPML.

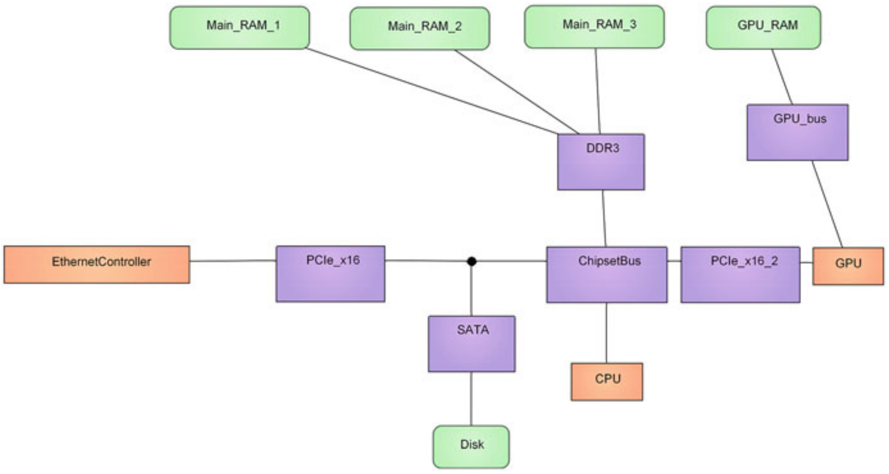


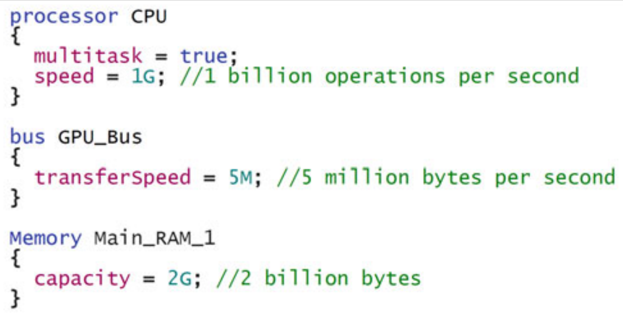
Fig. 7.12 A platform model in DPML

With just these blocks, we can create sufficiently realistic platform models for analysing the performance of a data path design. The platform of our running example looks like in Fig. 7.12.

Memory blocks are shown as rounded rectangles, bus and executor blocks as rectangles. Buses are the only components that can limit data transfer speed. Thus, the `Disk` memory, which models a hard drive, can in fact read and write data infinitely fast. The `SATA` bus models both the real SATA bus by means of which the hard drive is connected to the PC motherboard and the hard drive's inherent maximum read/write speed. Note that the model represents the main RAM and GPU RAM memories in the form of the (statically allocated) buffers `Main_RAM_1`, `Main_RAM_2`, `Main_RAM_3`, and `GPU_RAM`; the latter are the actual resources that tasks need to compete for.

A line between blocks in a platform model is called a **connection**, meaning that the two (or more) blocks are *directly* connected to one another. Connections limit the possible routes by which data can flow through the platform. An additional requirement is that, on a route between an executor block and a memory block, there may only be (one or multiple) bus blocks.

Resource blocks have properties, much in the same way as the steps of an application have behaviour. For example, properties of resource blocks include the bandwidth of a bus and the capacity of a memory. Analogously to steps and tasks or to objects and classes, the properties of resource blocks are specified in small chunks of code called **resources**. A resource describes that a particular piece of hardware exists and has some particular properties. For example, a resource may describe a particular Intel processor. A resource block based on that resource, describes that such an Intel processor is used in an actual hardware platform. A resource block cannot exist without an associated resource, and there can be multiple resource



```
processor CPU
{
  multitask = true;
  speed = 1G; //1 billion operations per second
}

bus GPU_Bus
{
  transferSpeed = 5M; //5 million bytes per second
}

Memory Main_RAM_1
{
  capacity = 2G; //2 billion bytes
}
```

Fig. 7.13 Some example resources

blocks based on a single resource. Resources are typically simpler in structure than tasks. Many resources only have one or two properties.

As shown in Fig. 7.13, each resource type has its own (small) set of properties. The `transferSpeed` property for buses and the `capacity` property for memories always have the same unit: they are expressed in bytes per second and in bytes, respectively.

DPML platform models only have buses, memories, and executors. This means there are no more specialised versions of such resources, such as caches or hard drives. It turns out that, currently, such elements are not needed.

Recall that a data path is a component that performs image processing and transfer operations. This allows us to make the following assumptions:

1. Data transferred between steps are *new*, i.e. a step has not recently read or written exactly the same image.
2. Data are read and written linearly, i.e. in a single stream of ordered bytes.
3. The amount of working memory needed for processing an image is small.

Assumptions 1 and 2 imply that caches do not influence processing speed when an image is read from memory, because every chunk of image data read constitutes a cache miss. Additionally, because of Assumption 3, we can assume that reads and writes to the internal memory used by the image processing steps (such as local variables) always constitute a cache *hit*, i.e. that they seldom go all the way to the actual memory block.

Because data are written and read linearly (Assumption 2), we can ignore the fact that physical hard drives are relatively slow when reading randomly offset data. Compared to the time spent reading relatively large chunks of sequentially stored bytes (image data), the *seek time* needed to move the read head to the right position is negligible.

Note that it is not impossible to design a data path in which one or more of the above assumptions do not hold. The analysis of such a DPML model may yield significantly different results than the real data path would. Therefore, it is important that DPML users are aware of these assumptions. If it turns out that some

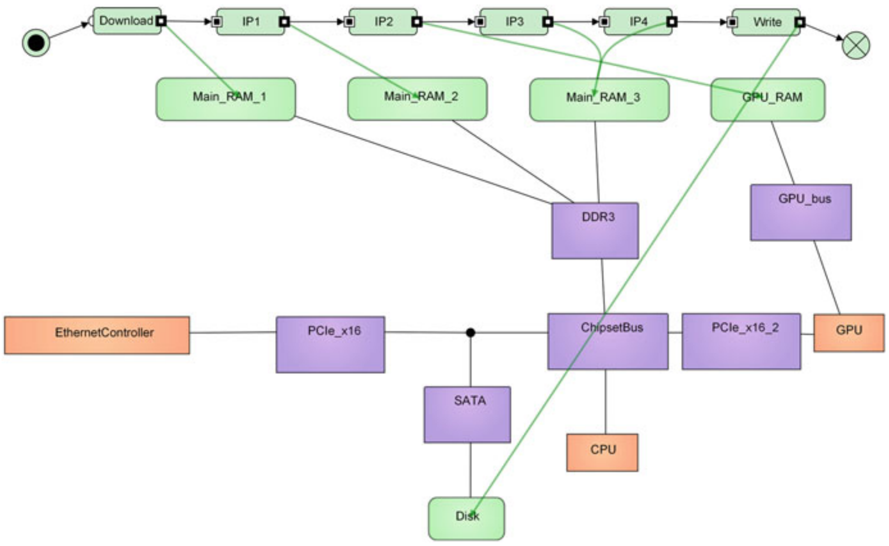


Fig. 7.14 Storage links in the mapping assign memory blocks to `Data` output pins

assumptions are invalid more often than not, additional features may be added to DPML to overcome this issue. Note that due to the modular and extensible structure of DPML, it is relatively easy to do so if the need arises. New properties for memory resources that describe, for instance, a hard drive's random seek penalty and its fragmentation state may be used to estimate seek times penalties if deemed significant. Similarly, there is no structural reason why a fourth resource type, such as a cache, could not be added to the language if necessary.

7.4.6 DPML: The Mapping View

A mapping defines how an application relates to a platform. In a mapping, we specify which steps run on which executor blocks, which data are stored where, and which memory claims and releases are performed. Like applications and platforms, mappings are partly textual and partly graphical.

DPML mappings have three different kinds of **links** by which elements are mapped onto one another: storage links, allocation links, and execution links. Visually, a mapping is simply displayed by an application and a platform shown alongside one another, and links are represented as arrows between the application and the platform.

Storage links specify where the output data on each `Data` output pin of a step have to be stored for a subsequent step to be able to read and process them. A storage link is thus a link between output `Data` pins and memory blocks. Figure 7.14 shows how we can model this in DPML for our running example.

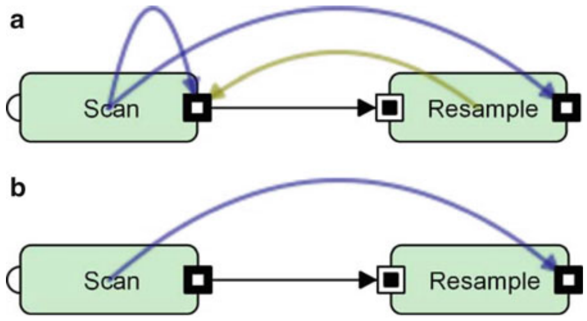


Fig. 7.15 Some allocation and release links (a) and their short-hand notation (b)

Allocation links are used to keep track of memory claims and releases. Before data can be written to memory, it must first be allocated. This is important, because if a step cannot start because memory is full, the step must be blocked until sufficient memory is available for its output `Data` pins.

There are two kinds of allocation links, see Fig. 7.15: claim links (blue) and release links (green). They are responsible for claiming and releasing the memory block, respectively. Even though links are part of the mapping, they are drawn entirely in the application; this is because instead of saying "step A claims memory B", we say "step A claims the memory needed for output pin C". Using the storage links, analysis tools can then determine which memory block that pin is mapped to, and using the properties of the task associated to the output pin's step, it can be determined how much memory should be claimed.

The allocation links may create a busy picture that is difficult to oversee. Therefore, DPML provides two shorthand rules: if a `Data` output pin has no claim link, then the step that the pin belongs to is assumed to perform the claim. Similarly, if it has no release link, then the following step is assumed to perform the release. The second rule can only be applied if there is only a single step that consumes the data produced by the `Data` output pin. If there are more than one (or zero) following steps, a release link should always be drawn.

Execution links, finally, describe which steps run on which executor blocks. Like storage links, they are drawn as simple arrows from steps to executor blocks. Every step can have at most one execution link, but an executor can be associated to any number of execution links.

Note that it is allowed for a step to not have an execution link, but only if the step has no `Data` pins. If a step consumes or produces data, then this means that data are being transferred from or to a memory, via some buses, to or from an executor block. Only by means of an execution link, this route can be computed.

Unlike storage links, each execution link has two additional properties: an associated *implementation* and a *priority*. Each (red) arrow between steps and executor blocks in Fig. 7.16 is an execution link.

An **implementation** describes *how* a step can be mapped onto an executor block. Because we are interested in the speed of the data path, an important property of a single step is its duration. A step's speed is typically limited either by the available

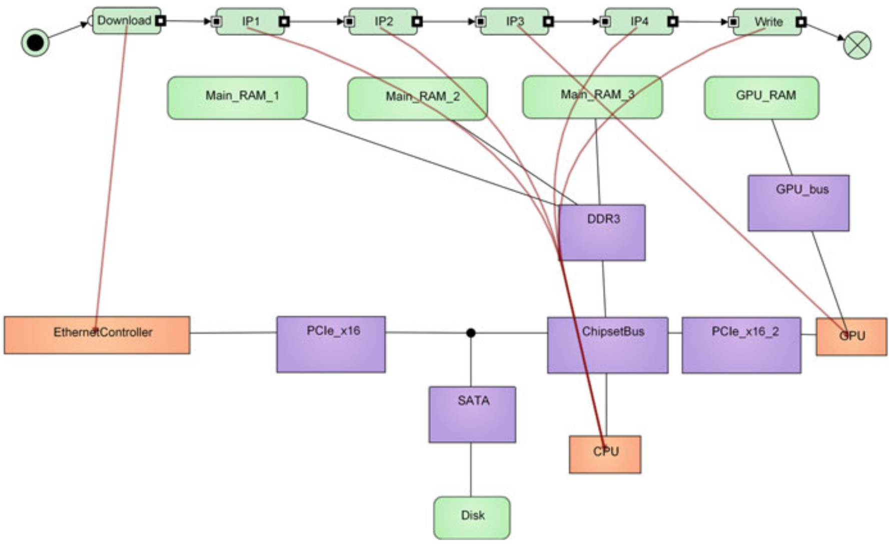


Fig. 7.16 Execution links

processing power, or by the available bus capacity for reading and writing its input and output data. In order to know which of the two limits a step's speed, we need to compute both. A step's bus usage can be derived from the size in bytes of its inputs and outputs and from the platform layout. Computing a step's processing speed, however, requires some more information from the user.

The duration of a single step depends on properties of the processor, on properties of the data (such as the image width and length) and on the particular implementation of the step. A DPML implementation captures this dependency in the property `processingDuration`. This property specifies the amount of time that a step is expected to take to process a single block of image data, given the current circumstances. As this time usually depends on some property of the input and/or output data as well as the amount of processor speed that is assigned to it, `processingDuration` is usually a function of these properties.

Figure 7.17 shows how a typical implementation for a resample task may look. This implementation models a situation in which the speed of resampling depends on the number of pixels of the largest image, which is the input image when scaling down, or the output image when scaling up. Moreover, on average, 20 clock cycles are used per pixel in the largest image. With this information, the `processingDuration` of the step can be computed. The implementation can directly refer to all properties of the associated task (such as the input pin and output pin properties).

A **priority**, formulated as an integer number, specifies which step gets to "go first" if multiple steps want to use a resource at the same time. We enforce one important convention with respect to priorities: the higher the number, the lower

```
implementation Resample_CPU performs Resample on CPU
{
  // We assume that a resample computation costs 20 clock
  // ticks per pixel, with the biggest of the two images
  // (depending on whether we zoom in or out) determining
  // the number of pixels

  cyclesPerPixel = 20;

  processingDuration =
    max(out.size, in.size) * cyclesPerPixel * processor.assignedSpeed;
}
```

Fig. 7.17 An implementation

the priority. So a step with priority 3 is considered more important than a step with priority 7. It is possible for multiple execution links to have the same priority, and this should imply that resources are fairly shared between the competing steps.

7.5 DSEIR: DSE Intermediate Representation

Section 7.3 motivated the importance of an intermediate representation to support DSE. We are developing the intermediate representation DSEIR specifically for the purpose of model-driven DSE. In modelling design alternatives, DSEIR follows the Y-chart paradigm. It further has support for defining experiments. It has been realised as a Java library, filling in the central module of the architecture of Fig. 7.4 on page 193. The current implementation supports four views: application, platform, mapping, and experiment. DSEIR can be used through a Java interface, an XML interface, and an Eclipse-based prototype GUI. This section introduces and illustrates the four views of DSEIR. We use the DSE case study of the previous section as a running example.

7.5.1 DSEIR: The Application View

Figure 7.18 shows a fragment of the DSEIR representation of the application part of our running printer example, in both the XML and the graphical format. The DSEIR application language is inspired by dataflow languages, in which data transformations play the most prominent role, but it intends to also support Petri-net and automata-based modelling concepts. An application consists of a number of tasks, and models the functional behaviour of the system. Each task has one or more ports, a number of load declarations and a number of edges. Ports are collections of values of some type (integer, integer array) and are either ordered in a fifo (first-in first-out) way or unordered. Ports provide inputs to tasks. The load declarations specify the load of the task for the *services* that it uses. These services are expected

a

```
<param name="NumPages" />

<global name="TIME_SCALE" value="1000" />
<global name="SCAN_CF" value="10" />
<global name="IP1_LOAD" value="300" />
<global name="A4_PIXELS_600_DPI" value="34802530" />
<global name="A4_RGB_BYTES_600_DPI" value="3*A4_PIXELS_600_DPI" />
<global name="NumBands" value="(A4_RGB_BYTES_600_DPI / (1024 * 1024 * 10)) + 1" />

<task id="Download">
    <port id="p" type="int" init="1" />
    <load service="TRANSFER" value="A4_RGB_BYTES_600_DPI/(SCAN_CF*1024)" />
    <load service="RESULT_STORAGE" value="1" />
    <edge port="p" value="p+1" cnd="p &lt; NumPages" />
    <edge port="p_b" task="IP1" value="[p,1]">
        <ho service="RESULT_STORAGE" value="1" />
    </edge>
</task>

<task id="IP1" cnd="p_b[0] == next_p">
    <port id="p_b" type="int[2]" order="unordered" />
    <port id="next_p" type="int" init="1" />
    <load service="COMPUTATION" value="(IP1_LOAD*TIME_SCALE)/NumBands)" />
    <load service="INTERNAL_STORAGE" value="1" />
    <load service="RESULT_STORAGE" value="1" />
    <edge port="p_b" value="[p_b[0], p_b[1]+1]" cnd="p_b[1] &lt; NumBands">
        <ho service="INTERNAL_STORAGE" value="1" />
    </edge>
    <edge port="next_p" value="if (p_b[1] &lt; NumBands) then next_p else next_p+1" />
    <edge task="IP2" port="p_b" value="p_b">
        <ho service="RESULT_STORAGE" value="1" />
    </edge>
</task>
```

b

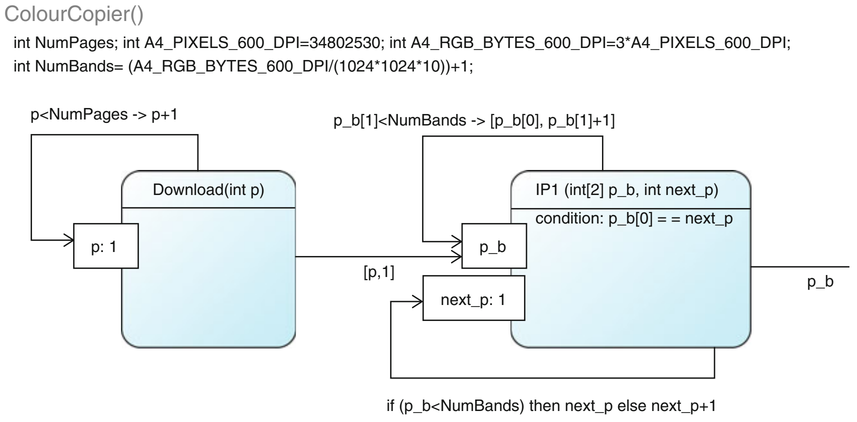

ColourCopier()
int NumPages; int A4_PIXELS_600_DPI=34802530; int A4_RGB_BYTES_600_DPI=3*A4_PIXELS_600_DPI;
int NumBands= (A4_RGB_BYTES_600_DPI/(1024*1024*10))+1;
p<NumPages -> p+1
Download(int p)
p: 1
[p,1]
p_b[1]<NumBands -> [p_b[0], p_b[1]+1]
IP1 (int[2] p_b, int next_p)
condition: p_b[0] = = next_p
p_b
next_p: 1
p_b
if (p_b<NumBands) then next_p else next_p+1

Fig. 7.18 An application in DSEIR: (**a**) XML; (**b**) graphical representation

to be provided by the platform (such as a COMPUTATION service that is provided by a CPU). The task loads are used in combination with platform information to determine the execution time of the task. The use of *service types* avoids direct references to platform resources which avoids coupling of the application description with a specific platform description. An edge leads from the current task to either a port of another task or to a port of the same task. The purpose of an edge is to add a new value to the target port. An edge has an *expression* in the DSEIR expression language that gives the new value in the target port. Furthermore, both a task and an edge can have a *condition*, which is a boolean expression that determines whether the task or edge is enabled and can execute. Finally, an edge can have zero or more *handover* specifications (the '`ho`' entries in the XML representation in Fig. 7.18) which contain an amount of allocated services that should be passed on to the next task. This allows modelling of resource reservations, for instance, memory pointers that are passed on from one task to the other without releasing the memory. An application can have a number of global variables of type integer, and a number of parameters, which also are of type integer. Both can be used in expressions in the application part. The DSEIR expression language is sufficiently powerful to capture applications at the intended abstraction level; it is kept as simple as possible though to facilitate model transformations to analysis tools.

Figure 7.18 shows the specifications for the Download and IP1 tasks of the running example. The XML representation includes the load and handover specifications, which in the figure are omitted from the graphical representation. Comparing the DPML model of the previous section with the DSEIR model, we see some differences. First of all, all concepts in DSEIR are independent of any specific application domain, whereas DPML intentionally contains elements from the domain of professional printing (with built-in concepts like pages, `PixelData` pins, and printer and scanner resources). Furthermore, the conversion from pages to bands is explicitly visible in the DSEIR task graph, whereas it is part of the implementation specifications in DPML. The most important difference, however, is the fact that DSEIR allows to specify task workloads and high-level resource management aspects in an abstract way in the application view, via services, loads, and handovers. This allows a strict decoupling between application and platform aspects, as further illustrated below. This fits with the goal of DSEIR as a domain-independent intermediate representation, which should allow to capture a wide diversity of mapping, scheduling, and resource allocation strategies. For a domain-specific language like DPML, predefined solutions for some of these aspects may be acceptable, which keeps the language simpler and more intuitive for domain engineers.

The semantics of a DSEIR application model is Petri-net like. A task can execute if all its ports have at least one value and if the condition of the task evaluates to *true* for the chosen port values. A task can choose any value from an unordered port or the first value from a fifo port. Upon execution start it consumes the chosen value from each port. When the task is finished, it executes all its enabled actions in an atomic fashion, which produces new values in a (sub)set of the target ports.

a

```
<resource id="cpu" capacity="4">
    <time service="COMPUTATION" value="1" />
</resource>

<resource id="ethernet" capacity="1">
    <time service="TRANSFER" value="(1000 * TIME_SCALE) / 70 * 1024)" />
</resource>

<resource id="b1" capacity="b1_size">
    <time service="INTERNAL_STORAGE" value="0" />
    <time service="RESULT_STORAGE" value="0" />
</resource>

<resource id="b2" capacity="b2_size">
    <time service="INTERNAL_STORAGE" value="0" />
    <time service="RESULT_STORAGE" value="0" />
</resource>
```

b

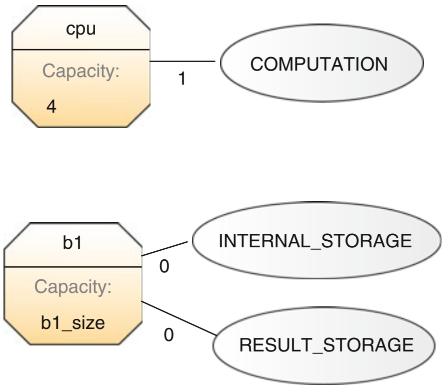


Fig. 7.19 DSEIR resource definitions in XML and graphical format

7.5.2 *DSEIR: The Platform View*

A platform in DSEIR consists of a number of resource declarations. Figure 7.19 shows a fragment of the DSEIR platform representation for our running printer example. Each resource has a capacity which can be read as the number of available units. Furthermore, a resource provides a number of *services* that tasks can use. A resource has a certain *service time* for each service it provides. This service time equals the number of time units that is needed to process one unit of load. This is used in combination with the load of a task to compute the task's execution time. The platform in the example has a quad-core CPU resource that provides a COMPUTATION service; the buffers provide INTERNAL_STORAGE and RESULT_STORAGE services, which allows to distinguish buffers for input and output data. The service time of the buffers is set to 0, which corresponds to an infinite processing speed. A platform can also have a number of parameters, which, like the application parameters, are of type integer. These can be used in expressions in the platform part, such as the capacity and speed expressions of resources. In the example, the buffer sizes are parameters (to be optimised in the DSE).

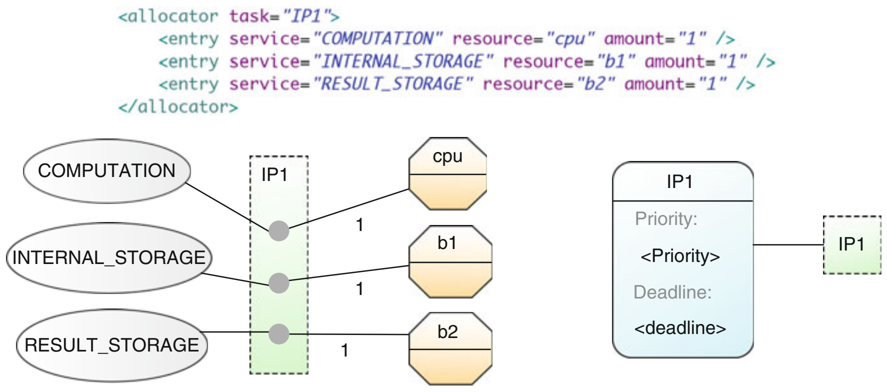
In line with the motivation for application-centric modelling laid out in Sect. 7.3 and in contrast to DPML models, a DSEIR platform model is simply a collection of resources without explicit structure. The structured platform models of DPML conform to the typical views of designers, whereas the unstructured models of DSEIR fit well with an intermediate representation that should be conceptually as simple as possible.

7.5.3 DSEIR: The Mapping View

The mapping ties an application to a platform, and consists of an *allocator* entry for each task, and *priority* and *deadline* specifications. An allocator element specifies to which resources the services required by the application are mapped. Furthermore, it specifies the amount of the resource that is allocated to the task. This amount should be less than or equal to the resource capacity. An allocator can be preemptive; by default it is non-preemptive. If it is preemptive then a running task can be preempted and the preemptive resource can be allocated to another task. If an allocator is non-preemptive and the available resource capacity is not enough for the task, then the task cannot run. Such resource-arbitration choices are made at time 0 (the start of system execution) and each time a task finishes. The priority elements specify the priority of tasks. By default, tasks have priority 0 (the lowest priority in DSEIR). The priority must be an integer expression and can depend on the run-time state, e.g. the port values for the current execution of the task. This allows full dynamic priority scheduling. Deadline specifications can be used for schedulability analysis (see [40]).

Figure 7.20 shows fragments of a DSEIR mapping representation. The visual representation allows to reuse allocators for multiple tasks. In the running example, however, each task has its own allocator, with the same name. This is because in this example each task asks for a unique combination of resources. Task IP1, for instance, is bound to allocator IP1, which provides INTERNAL_STORAGE through buffer b1, RESULT_STORAGE through buffer b2, and COMPUTATION through the CPU resource. Allocator IP2 (not shown in Fig. 7.20) binds buffer b2 for INTERNAL_STORAGE to task IP2, thus, accurately capturing the sharing of b2. Priorities are left unspecified, resulting in the default (lowest) priority of 0 for all tasks. Deadlines are also left unspecified, because they are not used in this example.

The examples given throughout this section show how the intended Y-chart separation of concerns between application and platform is achieved through a mapping view. The only interaction is through service definitions and allocators. Application and platform definition can be adapted fully transparently, as long as



```
<allocator task="IP1">
    <entry service="COMPUTATION" resource="cpu" amount="1" />
    <entry service="INTERNAL_STORAGE" resource="b1" amount="1" />
    <entry service="RESULT_STORAGE" resource="b2" amount="1" />
</allocator>
```

Fig. 7.20 Mapping in DSEIR

the service names do not change. The execution model of DSEIR is based on dynamic priority-based preemptive scheduling, which is a generic mechanism that allows designers to specify their own resource allocation and scheduling strategies through the allocator definitions. This fits with the needs of a generic intermediate representation. Considering the domain-specific language DPML, the resource allocation and scheduling mechanism is fixed in the native simulator, which on the one hand limits flexibility but on the other hand relieves designers from the task of specifying these aspects.

7.5.4 DSEIR: The Experiment View

DSE involves more than specifying design alternatives. It also requires the definition of experiments, among others. Experiments can also be described in DSEIR. Figure 7.21 shows an experiment definition. An experiment definition contains one or more experiment entries. If there is more than one, then these experiments are executed sequentially. An experiment entry has a name, that identifies the type of experiment, and can contain a *model* entry. The model entry can specify *one or more* models. Multiple models are specified using model parameters that may take different values. Finally, an experiment contains a number of properties. Every experiment type has its own set of properties with their own meaning. These experiment entries can be seen as invocations of predefined analysis recipes.

The example in Fig. 7.21 takes the models for our running example. The first experiment entry analyses all possible buffer size combinations for the buffers allocated in the main RAM memory, for a range of compression factors (parameters `minCF` and `maxCF`) and for 100 pages. It performs simulations (using CPN Tools [35], see next section) to explore the throughput (defined by the '`observers`' entry) that can be achieved for each combination of buffer sizes.

```
<experiment name="generate_traces">
    <model mapping="mapping.xml">
        <pvalue name="NumPages" values="100" />
        <pvalue name="minCF" values="10" />
        <pvalue name="maxCF" values="25" />
        <pvalue name="b1_size" values="1,2,3,4" />
        <pvalue name="b2_size" values="1,2,3,4" />
        <pvalue name="b4_size" values="1,2,3,4" />
    </model>
    <property name="observers" value="rt(sink)" />
    <property name="outputDir" value="traces" />
    <property name="number" value="10" />
    <property name="timeScaleDivision" value="1000000.0" />
    <property name="objectName" value="p" />
    <property name="launchResVis" value="false" />
</experiment>

<experiment name="extract_paralyzer_view">
    <property name="sourceDir" value="traces" />
    <property name="quantities" value="Timeunits per image = rt,
                RAM = b1_size * 25 + b2_size * 10 + b4_size * 6.2" />
    <property name="launchParalyzer" value="true" />
</experiment>
```

Fig. 7.21 An experiment in DSEIR

Per combination, ten simulations are performed (defined by the '`number`' entry). The other parameters set some values to format the output of the simulations. The second experiment entry takes the output and extracts a Pareto space that illustrates the trade-offs in the space, taking into account the variations that occur due to variation in compression factors. More details about the latter are given in the section presenting the diagnostic support in the Octopus tool set, Sect. 7.7.

The experiment view is an important part of DSEIR that allows designers to specify and maintain experiments. It is, for example, straightforward to re-run the same experiment on variants of a model. The tool set implementation, explained in some detail in Sect. 7.8, is such that it is easy to add new analysis plugins. An analysis plugin predefines an analysis recipe, as mentioned above, defining which tools are called, in which order, and with which parameter settings. This allows for example to easily add domain-specific analyses.

7.6 Analysis and Model Transformations

The previous two sections have introduced DPML, which served as an illustration of domain support that can be provided, and the intermediate representation, DSEIR. Together, DPML and DSEIR fill in the modelling perspective in our model-driven DSE philosophy and in the Octopus tool set. Design alternatives can be captured and it is possible to define experiments to explore the space of alternatives.

The experiments may perform various types of analysis on the specified design alternatives. This section presents *three types of analysis* supported in Octopus.

From an industrial perspective, *simulation* is the most important analysis technique. Simulation technology is mature and it may serve many different purposes, ranging from building a basic understanding of system behaviour, to detailed timing analysis and functional validation. The current tool set uses CPN Tools [35] for simulation of DSEIR models.

Another class of widely used analysis techniques are *model checking* techniques that are based on the underlying principle that a model is exhaustively analysed to conclude whether or not properties of interest hold for the model at hand. Octopus supports translation to the Uppaal [10] model checker. Uppaal offers (timed and untimed) model checking, which may be used to perform deadlock analysis, property checking, and timing and schedule optimisation.

Finally, for data-intensive applications, like print pipelines, performance and resource usage are often dominated by the flow of data through the system and the operations performed on these data; this is in contrast to control-intensive operations where communication and synchronisation typically determine the performance. Specialised *dataflow analysis* techniques allow fast exploration of design alternatives at a high level of abstraction. Octopus supports an experimental interface to the SDF3 [62] analysis tool for dataflow analysis.

7.6.1 Simulation with CPN Tools

Coloured Petri nets (CPNs) [36] are an expressive, precisely defined, and well-established formalism, extending classical Petri nets with data, time, and hierarchy. CPNs have been used in many domains (e.g. manufacturing, workflow management, distributed computing, and embedded systems). CPN Tools provides a powerful framework for modelling CPNs and for performance analysis (stochastic discrete-event simulation) on CPN models.

DSEIR as outlined in the previous section defines a syntax. An earlier version of DSEIR has a precisely defined semantics [70], defined by means of a structural operational semantics. This semantics elegantly separates the Y-chart aspects (application, platform, and mapping). It would be possible to provide also the current version with a semantics along these lines. However, since DSEIR also needs execution support, we have chosen to provide both the semantics and the execution support via a transformation to CPNs. The goal of the transformation from DSEIR to CPN Tools in the Octopus tool set is therefore twofold, namely (1) to precisely define the semantics of DSEIR, and (2) to provide execution support for the full DSEIR language.

Figure 7.22 illustrates the setup of the transformation from DSEIR models to CPNs. The basis of the transformation is a CPN template that contains the basic structure of the CPN model to be generated, the high-level dynamics of the resource handling, and monitors for producing simulation output. The template is filled with

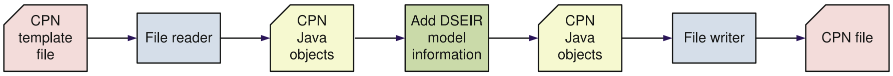


Fig. 7.22 Translating DSEIR specifications to CPN models

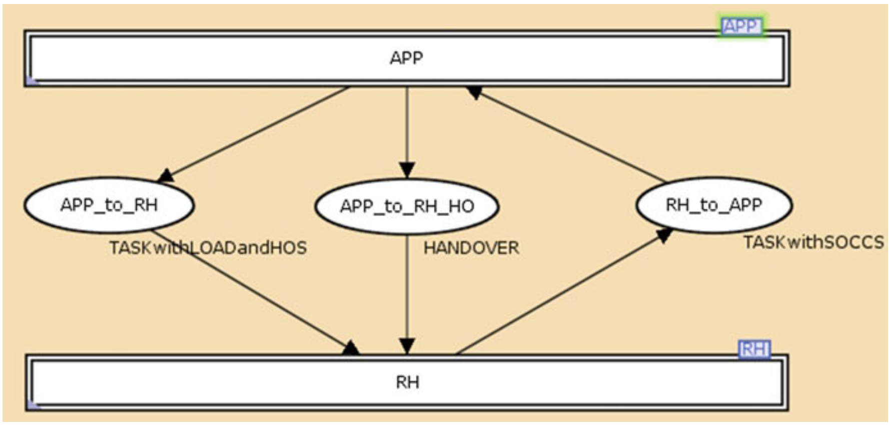


Fig. 7.23 Top-level view of the CPN model generated for the running example

the information from a concrete DSEIR model. The resulting CPN model can then be simulated by CPN Tools. Currently, there is an analysis recipe (see Sect. 7.8) that allows to simulate a specified number of runs of a given model. The execution traces resulting from these runs can then be further analysed, extracting properties such as the average throughput or resource utilisation, or observed bounds on performance properties such as latency or resource usage.

Figure 7.23 shows the top-level view of the CPN model generated for our running example. It illustrates the main structure of the CPN template used in the translation from DSEIR to CPN Tools. Any generated model consists of (1) the application view (block *APP*; a hierarchical transition in CPN terminology), (2) the specification of the resource handler (hierarchical transition *RH*), and (3) the interface between the two (components *APP_TO_RH*, *APP_TO_RH_HO*, and *RH_TO_APP*, called places in CPN terms). The information going from the application model to the resource handler contains the definition of a task to be started, the initial load of this task, and the handovers the task expects to receive. Information of the actual resource amounts (in terms of services) that a task occupies and notifications of a task being finished are communicated from resource handler to application.

Figure 7.24 shows the translation of the Download task, which is part of the application model generated for our running example. The task is split into a start event (transition *Download_s*) and an end event (transition *Download_e*). The first event sends task information to the resource handler (with the initial load specified in Fig. 7.18a) and puts a token into the waiting place *p_SE_Download*; the second event

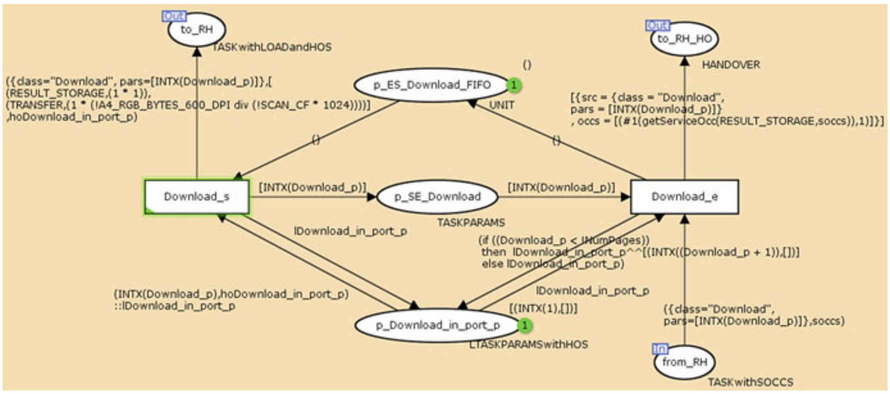
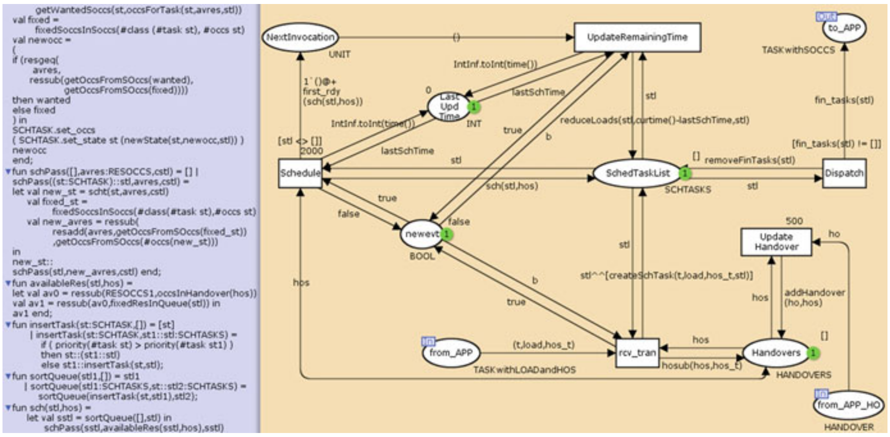

Out
to_RH
TASKwithLOADandHOS
({class="Download", pars=[INTX(Download_p)]},[
(RESULT_STORAGE,(1 * 1)),
(TRANSFER,(1 * (!A4_RGB_BYTES_600_DPI div (!SCAN_CF * 1024))))]
,hoDownload_in_port_p)
()
p_ES_Download_FIFO
1
UNIT
Out
to_RH_HO
HANDOVER
[{src = {class = "Download",
pars = [INTX(Download_p)]}
, occs = [(#1(getServiceOcc(RESULT_STORAGE,soccs)),1)]}]
()
()
Download_s
[INTX(Download_p)]
p_SE_Download
[INTX(Download_p)]
Download_e
TASKPARAMS
lDownload_in_port_p
(if ((Download_p < !NumPages))
then lDownload_in_port_p^^[(INTX((Download_p + 1)),[])]
else lDownload_in_port_p)
(INTX(Download_p),hoDownload_in_port_p)
::lDownload_in_port_p
lDownload_in_port_p
[(INTX(1),[])]
({class="Download",
pars=[INTX(Download_p)]},soccs)
p_Download_in_port_p
1
LTASKPARAMSwithHOS
In
from_RH
TASKwithSOCCS

Fig. 7.24 A fragment of the generated CPN application model for the running example



```
        getWantedSoccs(st,occsForTask(st,avres,stl))
  val fixed =
        fixedSoccsInSoccs(#class (#task st), #occs st)
  val newocc =
  (
  if (resgeq(
       avres,
      ressub(getOccsFromSOccs(wanted),
             getOccsFromSOccs(fixed))))
  then wanted
  else fixed
  ) in
  SCHTASK.set_occs
  ( SCHTASK.set_state st (newState(st,newocc,stl)) )
  newocc
  end;
▾fun schPass([],avres:RESOCCS,cstl) = [] |
  schPass((st:SCHTASK)::stl,avres,cstl) =
  let val new_st = scht(st,avres,cstl)
     val fixed_st =
          fixedSoccsInSoccs(#class(#task st),#occs st)
     val new_avres = ressub(
          resadd(avres,getOccsFromSOccs(fixed_st))
          ,getOccsFromSOccs(#occs(new_st)))
  in
  new_st::
  schPass(stl,new_avres,cstl) end;
▾fun availableRes(stl,hos) =
  let val av0 = ressub(RESOCCS1,occsInHandover(hos))
  val av1 = ressub(av0,fixedResInQueue(stl)) in
  av1 end;
▾fun insertTask(st:SCHTASK,[]) = [st]
      | insertTask(st:SCHTASK,st1::stl:SCHTASKS) =
        if ( priority(#task st) > priority(#task st1) )
        then st::(st1::stl)
        else st1::insertTask(st,stl);
▾fun sortQueue(stl1,[]) = stl1
     | sortQueue(stl1:SCHTASKS,st::stl2:SCHTASKS) =
       sortQueue(insertTask(st,stl1),stl2);
▾fun sch(stl,hos) =
     let val sstl = sortQueue([],stl) in
        schPass(sstl,availableRes(sstl,hos),sstl)
```

Fig. 7.25 The resource handler for the running example

occurs when the resource handler informs the application layer that the task has been finished. When this happens, the complete result storage is sent as a handover to the next task (see the annotation of the arc to place *to_RH_HO*) and an incremented page number is sent to place *p_Download_in_port_p*; the latter represents the self-loop of the Download task in Fig. 7.18b.

Figure 7.25 shows the internals of the resource handler generated for the running example; we briefly explain the logic of the resource-handling mechanism. Transition *UpdateHandover* ensures that arriving handovers are properly processed; this transition has the highest priority (500). Transition *rcv_tran* accepts new tasks and adds them into the queue of running tasks (place *SchedTaskList*). Transition *Dispatch* removes finished tasks from the queue and informs the application layer. The actual scheduling is done by the *Schedule* transition which has the lowest priority (2,000) and executes only if the queue is not empty. The function *sch* (shown on the left) modifies the task queue, according to the rules defined by the mapping specification. Place *NextInvocation* ensures that time progresses only to the first moment when some running task gets done. Transition *UpdateRemainingTime* updates the load of running tasks to reflect progress of time. Places *newevt* and *LastUpdTime* are auxiliary places to ensure a correct ordering of transitions. Note that the major part of the dynamics of the resource handler is premodelled and stored in the template. Only bodies of already specified functions are filled in when generating a CPN model for a concrete DSEIR specification.

The translation from DSEIR to CPN Tools is fast; also the simulations themselves are fast and scalable. However, CPN Tools compiles a model into an executable for performing the simulations. This compilation step is the slowest part in the transformation and analysis trajectory. For the exploration of large design spaces, in which many alternative models are simulated, this compilation step may become very time consuming.

7.6.2 Analysis with Uppaal

The timed automata formalism extends traditional finite-state automata with real-valued clocks [4]. This results in a concise formalism that is well suited to model state-based systems in which time plays a role. Properties of interest for such models can be phrased in temporal logic. Timed Computation Tree Logic (TCTL) is a logic that allows to specify properties with respect to the reachability of states within specified time bounds. This allows to specify, for example, that a page should be processed within a given latency, or that a certain error state should not be reachable. The fact that TCTL is decidable for timed automata [2] has led to the development of a number of analysis tools, *model checkers*, that compute whether a given timed automaton model satisfies a given TCTL specification. This section discusses the link from DSEIR to the Uppaal [10] model checker. The Uppaal input language extends the lean timed automata formalism with data (integer variables, language constructs to create C-like structures, etc.) and a C-like language to manipulate data. These features ease the creation and maintenance of models.

The translation from DSEIR to Uppaal is based on the following principles. First, every task in the application is translated to a separate Uppaal timed automaton, which has a clock to track the progress of the task. Tasks communicate through global variables that model the ports of the tasks. Second, resources are also modelled by global variables. Third, tasks read and write these in order to implement the allocation strategies as defined in DSEIR.

Figure 7.26 shows the Uppaal timed automaton for the Download task of the running example. The transition from *initialize* to *idle* initialises the port of the Download task with its initial value. The transition from *idle* to *active* models the start of the task. It picks the first port value (index *i*0) because the port is fifo.

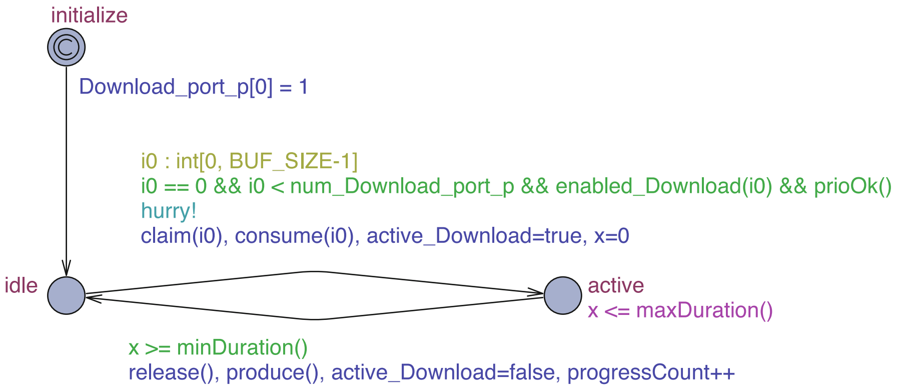

initialize
Download_port_p[0] = 1
i0 : int[0, BUF_SIZE-1]
i0 == 0 && i0 < num_Download_port_p && enabled_Download(i0) && prioOk()
hurry!
claim(i0), consume(i0), active_Download=true, x=0
idle
active
x <= maxDuration()
x >= minDuration()
release(), produce(), active_Download=false, progressCount++

Fig. 7.26 The Uppaal timed automaton of the download task from the running example

Furthermore, functions such as *claim* are called in order to do the bookkeeping with respect to resources, and the clock x is set to 0. The transition from *active* to *idle* models the completion of the task. The *release* function releases the resources and *produce* generates the values for the tasks' outgoing edges.

The main strength of the Uppaal model checker is that it enables *exhaustive* analysis of a DSEIR model. Currently, there are analysis recipes (see Sect. 7.8) to (1) check for deadlock situations, and (2) find precise bounds on resource usage (used memory and queue sizes, for instance) and latency. In addition, an experimental version of Uppaal-based schedulability analysis has been implemented (see [40]). These analysis recipes are only applicable to small to medium-sized models with limited non-determinism, because of the state-space explosion that is inherent in model checking. State-space explosion refers to the exponential growth of the state space with increasing model size. In this respect, model checking techniques contrast with simulation-based techniques which scale much better (but are not exhaustive).

Not all aspects of the DSEIR language are translatable to Uppaal. The main concept that cannot be translated directly is the concept of preemption. The DSEIR language is targeted at the system level of software-intensive embedded systems. This motivated the choice to allow DSEIR to approximate the progress of task execution by piece-wise linear behaviour. Consider, for instance, the situation that two tasks share the same processor core. The fine-grained division of time that may occur in reality is approximated by slowing both tasks down by some factor. This cannot be modelled by timed automata, although an approximation has been presented in [26, 29]. This approximation, however, fragments the symbolic state space built by Uppaal, which has a strong negative effect on the scalability of Uppaal analyses.

Two extensions of the Uppaal tool provide means to deal with preemptive behaviour more elegantly. First, Uppaal supports analysis of stopwatch automata (timed automata in which clocks can be stopped [14]) based on an over-approximation. This is useful to model situations in which a task is completely preempted.

This type of preemption is actually what is covered by the current translation from DSEIR to Uppaal; the aforementioned approximation is not supported. Second, the Statistical Model Checking (SMC) extension of Uppaal [13,18] features networks of priced timed automata, where clocks may have different rates in different locations. These networks of priced timed automata are as expressive as general linear hybrid automata [3]. SMC essentially provides stochastic discrete-event simulation for the combination of the timed automata and TCTL formalisms. The linear hybrid automata formalism is ideal for expressing the piece-wise linear progress of tasks as described above. The translation to this Uppaal variant, however, has not yet been realised because Uppaal-SMC has only been developed recently.

Besides the fundamental limitation with respect to preemptive behaviour, there are some practical limitations that have to do with the present Uppaal implementation: (1) the limited range of variables (32 bits for integers and 16 bits for clocks) sometimes leads to inaccuracies in approximating real values and to scaling problems in terms of the length and duration of executions being analysed; (2) the model state in Uppaal needs to be statically defined, which does not match well with the fact that DSEIR models do not have a priori bounds on the state (the port contents); this may lead to run-time errors that are typically hard to diagnose.

Supporting model transformations from DSEIR to multiple analysis tools raises the interesting question of consistency between these transformations. One may pick up the challenge to formally prove the correctness of a transformation with respect to the DSEIR semantics. An illustration of the type of proofs needed to show correctness of transformations can be found in [26, 71]. Those proofs were done in the context of an earlier version of DSEIR. We did not provide a formal correctness proof of the translation to Uppaal with respect to the CPN semantics for the current version of DSEIR. Pragmatically, when applying the CPN Tools and Uppaal analyses on models specified in the common subset of DSEIR supported by both translations, we get the same outcome.

7.6.3 Dataflow Analysis with SDF3

The model transformations discussed in the previous two subsections provide simulation support for the full DSEIR language and specialised analysis support for a subset of the language. The transformation from DSEIR to dataflow is of a different nature. The target of the transformation is Resource-Aware Synchronous DataFlow (RASDF) [76, 77], which extends the classical Synchronous DataFlow (SDF) [41] model of computation with a notion of resources in the style of the Y-chart. SDF has limited expressiveness, but this restriction allows more powerful analysis. It is for example possible to minimise memory requirements for a given throughput requirement [60, 78].

SDF models tasks by means of actors with a fixed execution time and tasks are assumed to communicate fixed amounts of data in all their executions. SDF therefore does not allow to capture dynamics such as variable execution times

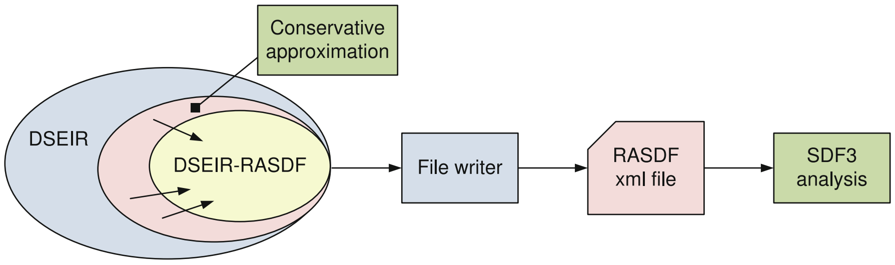


Fig. 7.27 The transformation from DSEIR to RASDF/SDF3

and communication rates explicitly. With conservative (worst-case) task execution times and communication rates, however, it is possible to determine bounds on performance and resource usage, such as the minimal throughput that can be guaranteed and the smallest amount of memory that suffices to guarantee that throughput. The model transformation from DSEIR to RASDF therefore aims to translate a subset of the DSEIR language to RASDF models that are conservative in terms of performance and resource usage. Figure 7.27 illustrates the transformation.

The figure shows that it is possible to directly transform a restricted subset of DSEIR, DSEIR-RASDF, to RASDF. DSEIR-RASDF is the subset that simply limits DSEIR to RASDF models. Using static analysis of RASDF models, it is possible to enlarge the subset of DSEIR models that can be conservatively captured as a DSEIR-RASDF model. DSEIR-RASDF models can then be exported to the SDF3 tool [62] for analysis, allowing the already mentioned throughput and resource usage analysis. More details about this transformation can be found in [48].

The described approach can only be applied to models with limited variation in task execution times and communication rates. If the variation in execution times and communication rates is too large, such as for example the variations because of compression rates in the running example (see Sect. 7.7), then the conservative bounds on the throughput and memory requirements that can be guaranteed are too loose to be practically useful. As a future extension, it will be interesting to investigate a link to Scenario-Aware DataFlow (SADF) [67, 79]. SADF allows to capture a finite, discrete number of workload scenarios, each characterised by an SDF model. A workload scenario may correspond to for example different types of pages (text, image) to be printed. Scenario transitions can then be captured in a state-based model such as a finite state machine or a Markov chain. Many analysis techniques for SDF can be generalised to SADF [61], which therefore provides an interesting compromise between expressiveness and analysis opportunities.

7.7 Diagnostics

The Octopus tool set has two tools for visualisation and diagnostics. The ResVis tool, short for *Res*ource *Vis*ualisation, see [59], can be used for detailed analysis of

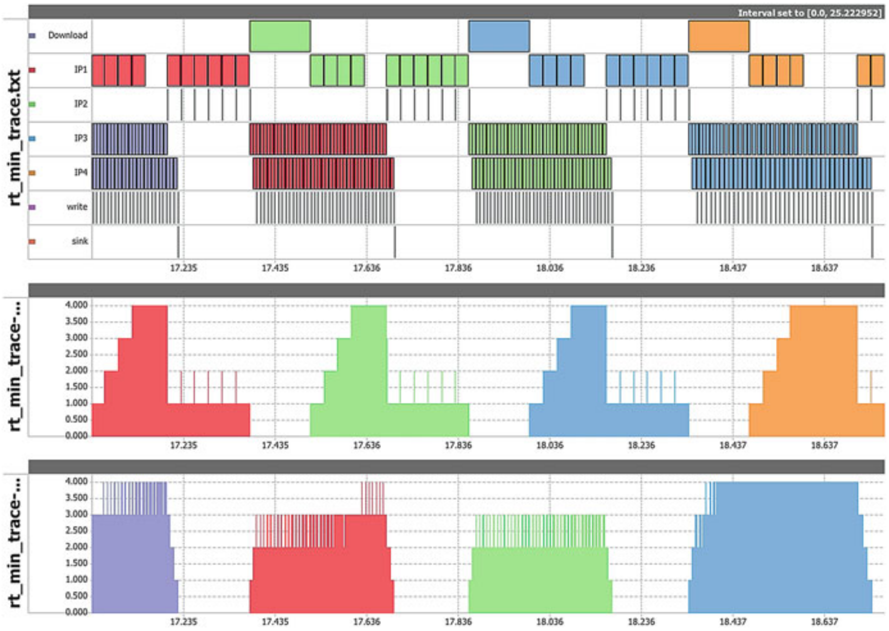


Fig. 7.28 Gantt chart and resource plots of buffers b2 and b4 of the running example

the behaviour of individual design alternatives. The Paralyzer tool, short for *Par*eto an*alyzer*, see [23], on the other hand, provides Pareto analysis and supports trade-off analysis between different design alternatives. The tools thus are complementary to each other. They fill in the diagnostics module of Fig. 7.4 on page 193.

7.7.1 Visualisation and Analysis of Execution Traces

The ResVis tool can be used to visualise individual executions of the modelled system by means of a Gantt chart. A Gantt chart shows the task activity and/or resource usage over time. Figure 7.2 on page 192 shows a ResVis Gantt chart for part of an event trace from one of the case studies performed with Océ-Technologies. Figure 7.28 shows a part of an event trace (top) for the running colour copier example and the accompanying resource plots of buffers b2 and b4 (middle and bottom, respectively). The model for the running example reserves one slot for b1 and four slots for b2 and b4. The event traces show the execution of the individual tasks. The resource plots show the resource usage of individual resources. The tasks and resource usage blocks can be coloured according to one of some predefined schemes such as by page number, use case (e.g. printing, scanning), or job (e.g. printing a specific number of pages of a given type). This is specific for the printer domain. The colouring in Figs. 7.2 and 7.28 is by page number.

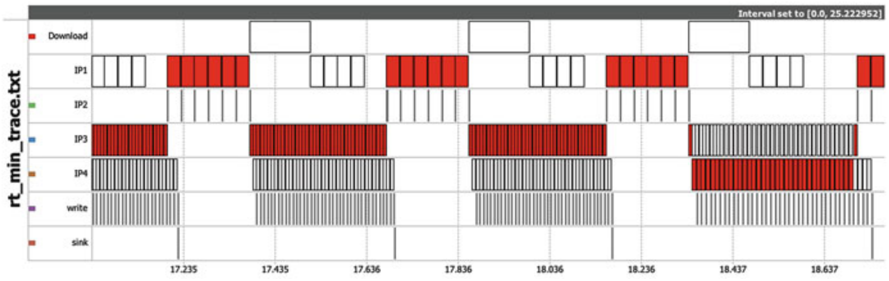


Fig. 7.29 The critical tasks in the execution trace of Fig. 7.28 are highlighted

Event traces and resource plots are very suitable to study the detailed dynamic behaviour of a design. They show for example implicit dependencies between tasks such as unexpected blocking and utilisation of resources. A typical use is to find bottleneck tasks and bottleneck resources. These are tasks and resources that determine the performance of the system. The Octopus tool set contains algorithms to compute an over-approximation of the set of critical tasks to support this type of analysis [24].

For instance, Fig. 7.29 shows the critical tasks of the trace shown in Fig. 7.28. Visualisation of critical tasks can ease the identification of bottleneck tasks and resources. Figure 7.29 shows that IP1, IP2, IP3, and IP4 form the critical path. (The colouring is not visible for the very fast task IP2.) Note that the application as shown in Fig. 7.9 has no data dependencies from IP3 to IP2 and also not from IP2 to IP1. Yet, the critical tasks show that IP1 must sometimes wait for IP2, and that IP2 must sometimes wait for IP3. This is caused by full buffers between these tasks, which limits parallelism.

Figure 7.30 shows critical tasks after an increase of b2 to 10 (top graph) and after an additional increase of b1 to 2 (bottom graph). Both changes positively affect the throughput of the system. After the last increase, there is little or no room left for further improvement. The critical path visualisation is a great help for solving problems with respect to time-related performance issues.

7.7.2 *Trade-Off Analysis with Uncertain Information*

Design-space exploration is a multi-objective optimisation problem. On the one hand, there is the design space consisting of all possible design alternatives. On the other hand, there is the cost space with its multiple cost dimensions, such as throughput, total memory usage, etc. Every design alternative is linked to the cost space, and the question is to find the best design solutions with respect to the considered cost dimensions. Pareto analysis is a well-known way to deal with this [53]. The Octopus tool set uses the Paralyzer library to perform Pareto analysis,

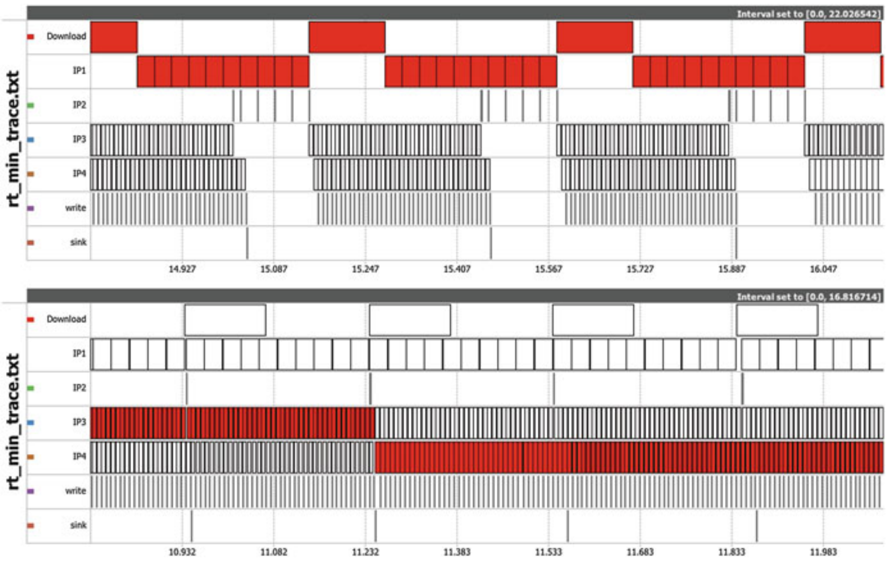


Fig. 7.30 The critical tasks after increasing the number of slots of b2 to 10 (above), and after an additional increase of the number of slots of b1 to 2 (below)

including the application of constraints and cost functions, and a means to visualise the trade-offs in two or more dimensions [23]. Furthermore, it can cope with uncertainty, which is typically present in the early phases of system design, by associating a design alternative with *sets* of points in the cost space, not with just a single point.

Consider the running example introduced in Sect. 7.4.1. The DSE question asked is: Minimise the amount of memory allocated to the buffers while retaining a minimum given throughput. In order to answer this question, a DSEIR model was created, which is parameterised with the three buffer sizes; fragments of this model are shown in Sect. 7.5. The possible values for the buffer sizes are taken from the set $\{1, 2, \ldots, 10\}$, which gives a design space of $10 \times 10 \times 10 = 1{,}000$ design alternatives. The Octopus tool set has been used to compute the throughput and the memory consumption of this set of design alternatives. The exact analysis through the Uppaal model checker and the analysis recipe for analysing bounds as mentioned in the previous section turns out to be too slow and the state space becomes too large because of the workload variations (different compression factors) and the number of pages in a job. Therefore, 30 simulation runs per configuration were made. The results are shown in Fig. 7.31.

The graph shows the cost space of the DSE question. The y-axis shows memory used for buffering; the x-axis shows throughput. The inverse of throughput is used so less is better. Each coloured rectangle represents a Pareto-optimal design alternative. This is a design alternative which is not dominated by any other design alternative,

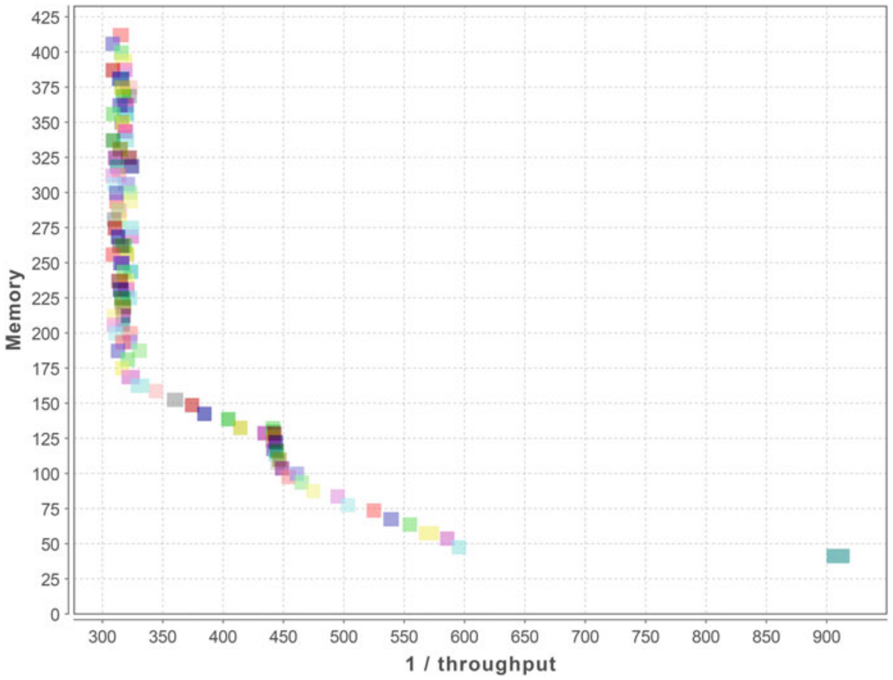


Fig. 7.31 Trade-off view of the total buffer size versus the throughput for the running example

where a design alternative dominates another one if it is not worse in any dimension of the cost space and better in at least one dimension. For instance, the blue rectangle at the right-hand side represents the design alternative in which all buffers have size 1. This is the cheapest design alternative in terms of memory usage, but its throughput is low. Note that design alternatives are associated with a subset of the cost space and not with a single point. Each design alternative has variation in its throughput caused by variations (stochastic behaviour) in the input (not every image is the same, leading to different compression rates). This variation is visualised by using rectangles of various sizes instead of points in the graph. Because of the use of simulation, the throughput bounds for each of the rectangles are only approximations of the true bounds. The graph shows that the throughput increases with larger buffers. However, using more than 175 units of memory does not result in a significant further increase in performance. The many configurations with more than 175 units of memory are on the Pareto front because they overlap in the throughput dimension. They are therefore not dominated by other configurations. This is a consequence of the fact that design alternatives are associated with sets of points in the cost space instead of with a single point.

7.8 Implementation Aspects

The Octopus tool set currently consists of various separate applications: the domain-specific data path tooling introduced in Sect. 7.4, the ResVis and Paralyzer diagnostics tools, which have been discussed in the previous section, and the generic Octopus application which is discussed in this section. The main aim of the Octopus application is to provide a formal, flexible, and extensible infrastructure to efficiently solve DSE problems that have been modelled in the DSEIR modelling language described in Sect. 7.5.

A DSE problem, by definition, has a number of design alternatives that need to be analysed. The analysis runs for different design alternatives are typically independent (although it is an interesting topic for further research to investigate incremental DSE techniques where analysis results for one design alternative can be reused for other alternatives). Evaluation of design alternatives can be distributed with little effort over a possibly heterogeneous set of computational nodes. The Octopus implementation has built-in support to automatically distribute analyses over computational nodes and collect results from these analyses. Almost any modern computer has multiple processing cores, so this support is very useful in practice. The simulations performed for the running example of the high-end colour copier reported on in the previous section show that distribution indeed can be very effective. The computation time decreases almost linearly with the number of available computational nodes.

The extensibility requirement for the Octopus tool set, together with the wish for platform independence and distribution support have led to the decision to build Octopus upon the OSGi runtime [52] environment. This Java-based module framework facilitates extensible service-oriented architectures. An important feature is the service registry which enables publication and lookup of services (implementations of Java interfaces). The modules are so-called OSGi bundles, which are plain JAR files. The OSGi framework enables dynamic addition, update, and removal of bundles and of services. Typically, bundles use other services to implement their own service interface. This service orientation often results in loosely coupled and easily testable components. The OSGi implementation that is currently used is Apache Felix [5]. The Octopus workflow treats both models and analysis results as data that can be transformed and visualised. The Octopus implementation architecture and data-centric approach is similar to that of the macroscopic tools [12] and, more specific, the CIShell [16] as described by Börner.

Figure 7.32 shows the high-level architecture of the Octopus implementation. It consists of the following components:

- The *Octopus API* (application programming interface) consists of several parts. It contains types to create DSEIR models, either programmatically in Java or from an XML description. Furthermore, it contains types for analysis results, such as a generic execution trace format. It also contains the service interfaces of the Octopus platform services that allow developers to program tool and analysis plugins, as discussed below.

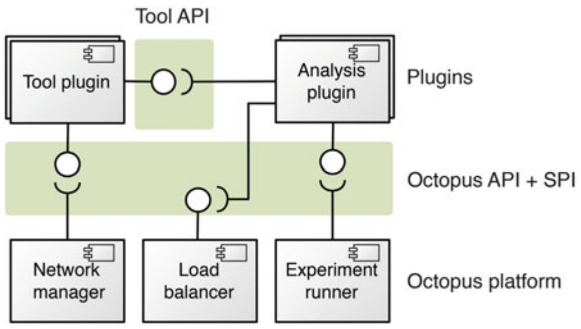


Fig. 7.32 High-level architectural overview of the Octopus implementation

- The *Octopus SPI* (service provider interface) contains the service interfaces that plugin developers should implement and register in the OSGi framework to extend the functionality of the tool set with a new analysis tool.
- The *Octopus platform* currently contains components that are necessary to realise the API and SPI functionality. Most notably, the *LoadBalancer* and *NetworkManager* realise the distribution capabilities of the tool set explained above, and the *ExperimentRunner* provides a user interface to run experiments.
- *Tool plugins* are OSGi bundles that register implementations for certain SPI types. They facilitate the use of dedicated analysis tools and typically implement a model transformation from DSEIR to the input language of the supported tool, and a transformation from the tool output to a general format specified in the Octopus API such as the trace format for execution traces. Tool plugins are free to specify their own API, as different tools can have very different functionality. Currently, as discussed in Sect. 7.6, there are mature plugins for CPN Tools [35] and Uppaal [10], and an experimental plugin for the dataflow analysis tool SDF3 [62].
- *Analysis plugins* provide a means for the user to use the tool plugins, and they thus typically use the APIs of the tool plugins and the Octopus API. These plugins form the analysis recipes that can be used in the DSEIR experiment view. They implement part of the SPI, in order to register themselves in the Octopus framework. The current version of Octopus has the following analysis plugins:
  - *GenerateTraces* uses the CPN Tools plugin to generate a specified number of execution traces for a given non-empty set of models, and collects the output in the form of execution traces. This plugin was used to generate the traces underlying the Pareto analysis shown in Fig. 7.31. Individual traces can be visualised by the ResVis diagnostics tool.
  - *VerifySystem* uses the Uppaal plugin to verify that the system satisfies some sanity properties such as, for example, that there is no deadlock.
  - *RandomUppaalTrace* uses the Uppaal plugin to generate a random trace. It uses a built-in Uppaal facility for generating execution traces, and makes this available to the Octopus user. In general, the GenerateTraces plugin is faster though.

– *GenerateBounds* uses the Uppaal plugin to compute lower and upper bounds on application latency, resource usage, and port usage (the number of items present in any of the ports in a DSEIR application model).
– *GenerateParalyzerView* collects performance data created by any of the aforementioned analysis plugins in order to generate a Pareto trade-off view as shown in Fig. 7.31.
– *CriticalPathAnalyzer* applies critical-path analysis to a specific trace file. The results can then be visualised in ResVis, as illustrated in Figs. 7.29 and 7.30.

7.9 Industrial Experiences

We have used Octopus in four case studies at Océ-Technologies. These case studies all involve design-space exploration of printer data paths of professional printers.

7.9.1 High-Performance Production Black-and-White Printing

Figure 7.33 shows an abstracted view of an FPGA-based data path platform of a high-performance production black-and-white printer, that supports use cases such as printing, copying, scanning, scan-to-email, and print-from-store. All required image processing algorithms are realised in the main FPGA (the IP blocks in the figure); the FPGA is connected to two memories via a memory bus. The machine

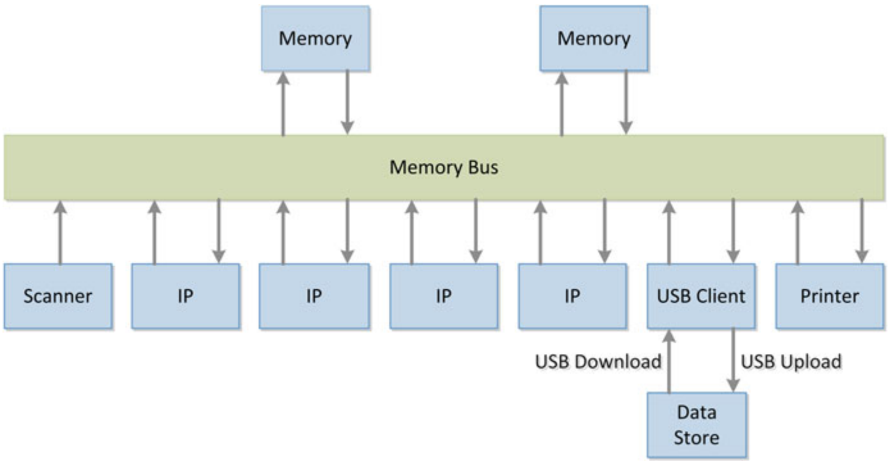


Fig. 7.33 An abstracted view of the platform of a high-performance production *black*-and-*white* print and scan data path (figure from [30])

can be accessed locally through the scanner and both locally and remotely through a print controller. Print jobs enter the system through the data store shown in the figure. The use cases all use different combinations of components in the platform. A print job arriving from the Data Store undergoes several image processing steps, with intermediate results being stored in one of the memories, after which the processed result is both sent to the Printer block and stored in the Data Store. The latter is useful for error recovery and for printing multiple versions of the same document. A scan job uses the Scanner board and several IP blocks, with intermediate results stored in one of the memories and the final result stored in the Data Store. Scanning and printing can execute in parallel, and also within the scan and print image processing pipelines, tasks may be executed in parallel. Resources like the memory and the associated memory bus, as well as the USB are shared between tasks and between print and scan jobs running in parallel. Moreover, the available USB bandwidth dynamically fluctuates depending on whether it is used in one or in two directions simultaneously.

Given the characteristics of the use cases and the FPGA-based platform, the data path in this case study can be modelled quite accurately with fixed workloads (for the various processing tasks, the memory bus, and the memories) at the abstraction level of pages. Only the USB client shows variation due to the above-mentioned variation in available bandwidth between unidirectional and bidirectional use. Schedule optimisation using Uppaal analysis is therefore feasible. USB behaviour can either be approximated with a fixed bandwidth or with a discrete approximation of the fluctuating behaviour as explained in Sect. 7.6.2. The models we developed were used for determining performance bounds and resource bottlenecks, for analysing interaction between scanning and printing, and for exploration of scheduling priorities and resource allocation (memory allocation, page caching) alternatives. One of the concrete results was an improved task prioritisation for the static priority scheduling employed in the platform. Further, Fig. 7.34 shows the Gantt chart of simultaneously running scan (red) and print (blue) jobs. The scan job is disrupted, because printing has priority on the USB. It only continues after the print job has finished. This behaviour materialises because the model lacks a crucial scheduling rule, stating that uploads over the USB are handled in the order of arrival, irrespective of the origin (the scan or print job) of the upload. This scheduling rule is actually enforced in the print controller, which is a component external to the data path. The analysis shows the importance of this external scheduling rule.

An interesting conclusion with respect to the Octopus tool set is that automatically generated Uppaal models are in comparison better tractable than the handcrafted models reported on in [29]. Longer print jobs can be analysed, due to a reduction in memory needed by Uppaal. Analysis times increase though. Memory usage and analysis times strongly depend on the size of the print jobs being analysed. For latency optimisation of two simultaneously running jobs, memory usage and analysis time range from kilobytes and seconds for small jobs of a few pages to gigabytes and hours for large jobs with hundreds of pages. Details can be found in [26].

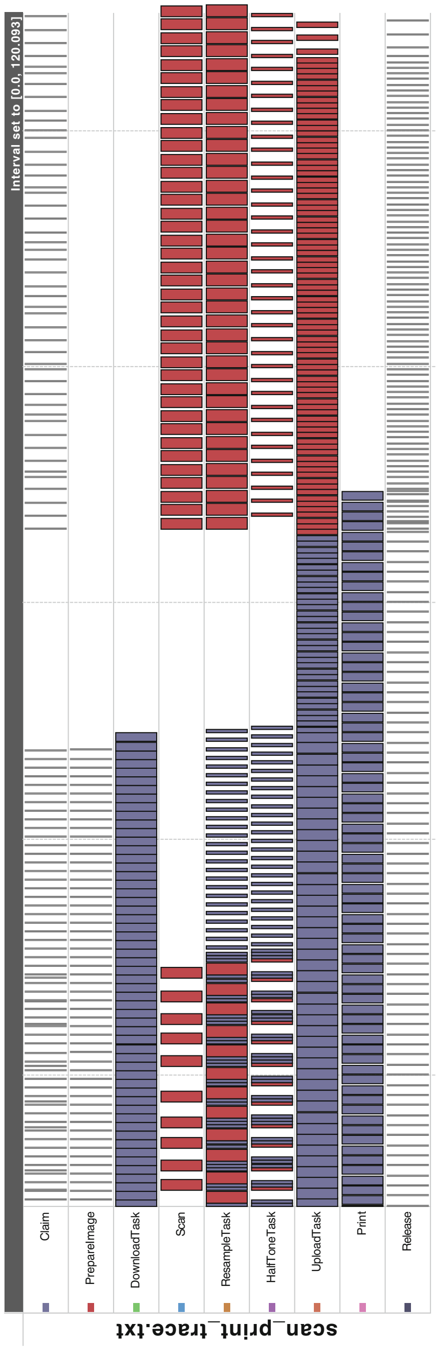


Fig. 7.34 The Gantt chart of simultaneously running scan (*red*) and print jobs (*blue*)

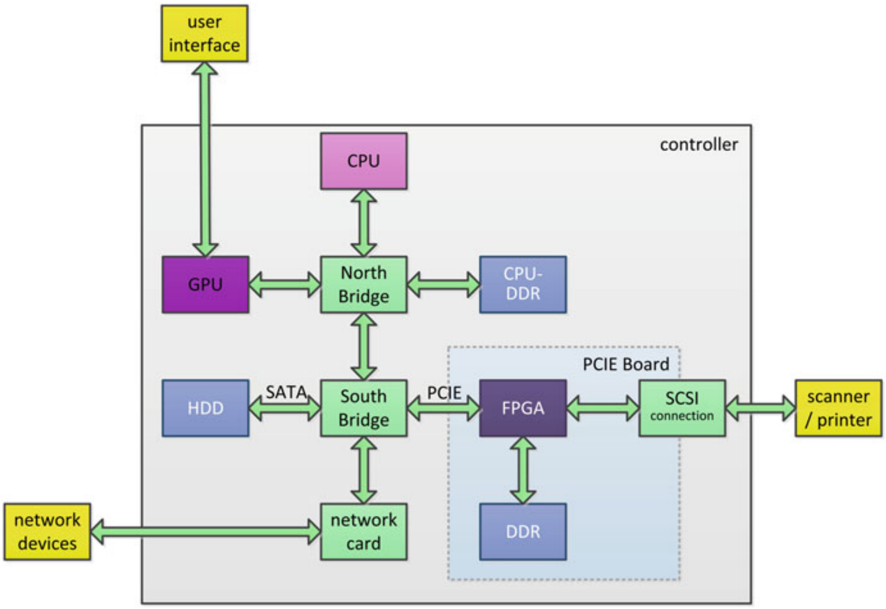


Fig. 7.35 A data path platform template used in several high-end colour copiers (figure from [8])

7.9.2 *High-End Colour Printing*

The other three case studies involved variants of a family of high-end colour copiers. Figure 7.35 shows a template of the data path platform used for these printers. It is a heterogeneous multi-processor platform that combines one or more CPUs (running Microsoft Windows) with a GPU, one or more Harddisks (HDDs), and an FPGA. Because of heterogeneity and the use of general CPUs, capturing the platform in a high-level abstraction is more challenging than modelling the platform of the first case study. The variation in workloads due to compression and decompression steps in the print and scan pipelines adds complexity, as well as the fact that pages are broken into bands and sometimes even lines to increase pipelining opportunities in the image processing pipeline. The latter is needed because high-resolution colour printing and scanning involves much larger volumes of data than black-and-white printing and scanning. The Gantt chart shown in Fig. 7.2 is in fact taken from one of these three case studies, and illustrative for the mentioned challenges. Also the running example is taken from one of these case studies.

The complexity of the models for the colour printer data path case studies is such that only simulation is sufficiently fast to do any practically meaningful analysis. For the first one of these cases, we started out with handcrafted models. Later we made DSEIR models, and for the last case study also DPML models. Automatically generated CPN models turned out to yield simulation times similar to the handcrafted CPN models. Simulation times range from seconds to minutes, depending on the size of the jobs being simulated. The translation of a DSEIR model to a CPN takes typically less than a second. The time that CPN Tools needs to compile a (handcrafted or generated) CPN model into a simulation executable is in the order of tens of seconds. Simulation times for DPML models with the native DPML simulator are of the same order of magnitude as CPN Tools simulation times.

Our analyses identified performance bounds for the print and scan pipelines and resource bottlenecks, and they were used to explore the interaction between scanning and printing. Buffer requirements were analysed and potential savings in buffering were identified. The impact of several changes in the image processing pipelines were analysed before the changes were realised, and a sensitivity analysis was performed for task workloads that were not precisely known.

The three colour copier case studies showed that the Octopus tools can successfully deal with several modelling challenges, like heterogeneous processing platforms with CPUs, GPUs, FPGAs, and various buses, varying and mixed abstraction levels (pages, bands, lines), preemptive and non-preemptive scheduling, and stochastic workload variations (due to input variation and caching). The mixing of abstraction levels in a single model was crucial to obtain the appropriate combination of accuracy and acceptable simulation speed.

7.9.3 General Lessons Learned

We can draw some general conclusions from the performed case studies. DPML and DSEIR allow to capture industrially relevant DSE problems. Both DPML and DSEIR models can be made with little effort, similar to the time investment needed for a spreadsheet model. Because of the provided modelling templates, creating a DPML or DSEIR model takes much less effort than creating a CPN Tools or Uppaal model. An important advantage of the use of an intermediate representation is that one model suffices to use different analysis tools, which means a further, substantial reduction in modelling effort when compared to handcrafting models for multiple tools. Model consistency is moreover automatically guaranteed. The aspect that is in practice the most tedious and time-consuming part of the modelling are the task workload models (processing, bandwidth, and storage requirements). These workload models are typically estimates based on experience of engineers or based on profiling measurements on partial prototypes or on earlier, similar machines. Note that these workload models are typically independent of the chosen modelling approach. Spreadsheet models, DPML, DSEIR, CPN Tools, and Uppaal alike need the same workload models as input. An important positive observation from the case studies is that the involved designers all reported a better understanding of the systems. Ultimately, the DSE models are envisioned to play an important role in documenting a design.

7.10 Discussion and Conclusions

This chapter has presented the Octopus view on model-driven design-space exploration (DSE) for software-intensive systems, elaborating on the DSE process and envisioned tool support for this process. Model-driven DSE supports the systematic evaluation of design choices early in the development. It has the potential to replace or complement the spreadsheet-type analysis typically done nowadays. Model-driven DSE can thus reduce the number of design iterations, improve product quality, and reduce cost.

To facilitate the practical use of model-driven DSE, we believe it is important to leverage the possibilities and combined strengths of the many existing languages and tools developed for modelling and analysis, and to present them to designers through domain-specific abstractions. We therefore set out to develop the Octopus DSE framework that intends to integrate languages and tools in a unifying framework. DSEIR, an intermediate representation for DSE, plays a central role in connecting tools and techniques in the DSE process in a flexible, extensible way, encouraging reuse of tools and of models. DSEIR allows to integrate domain-specific modelling, different analysis and exploration techniques, and diagnostics tools in customisable tool chains. Through DSEIR, model consistency and a consistent interpretation and representation of analysis results can be safeguarded. The current prototype tools combine simulation and model checking in the Octopus framework. DPML, a domain-specific modelling language for printer data paths, has been developed to provide support for the professional printing domain. The first industrial experiences with the Octopus approach in the printing domain have been successful.

Several challenges and directions for future work remain, both scientific challenges and challenges related to industrial adoption of model-driven DSE.

First of all, DSEIR needs further validation, also in other domains. It is already clear that extensions are needed, so that it covers all aspects of the DSE process. In particular, besides design alternatives and experiments, it would be beneficial to standardise the language for phrasing DSE questions and the format capturing DSE results. This would further facilitate the exchange of information between tools and the consistent interpretation of results.

Another direction for future research are the model transformations to and from DSEIR. The three transformations to analysis tools presented in this chapter are all of a different nature. It is important to precisely define the types of analysis supported by a transformation, the properties preserved by the transformation, and its limitations. Also techniques to facilitate (semi-)automatic translations and maintenance of transformations are important, to cope with changes in the Octopus framework or the targeted analysis tools.

Given precisely defined modelling languages and model transformations, it becomes interesting to explore integration of analysis techniques. Can we effectively combine the strengths of different types of analysis, involving for example model checking, simulation, and dataflow analysis? What about adding optimisation techniques such as constraint programming and SAT/SMT solving? No single analysis technique is suitable for all purposes. Integration needs to be achieved without resorting to one big unified model, because such a unified model will not be practically manageable. But how do we provide model consistency when combining multiple models and analysis techniques?

On a more fundamental level, integration of techniques leads to the question how to properly handle combinations of discrete, continuous, and probabilistic aspects. Such combinations materialise from combinations of timing aspects, user interactions, discrete objects being manipulated, physical processes being controlled, failures, wireless communication, etc.

Scalability of analysis is another important aspect. Many of the analysis techniques do not scale to industrial problems. Is it possible to improve scalability of individual techniques? Can we support modular analysis and compositional reasoning across analysis techniques, across abstraction levels, and for combinations of discrete, continuous, and probabilistic aspects?

Early in the design process, precise information on the workloads of tasks to be performed and on the platform components to be used is often unavailable. Environment parameters and user interactions may further be uncontrollable and unpredictable. How can we cope with uncertain and incomplete information? How do we guarantee robustness of the end result of DSE against (small) variations in parameter values? Can we develop appropriate sensitivity analysis techniques?

The increasingly dynamic nature of modern embedded systems also needs to be taken into account. Today's systems are open, connected, and adaptive in order to enrich their functionality, enlarge their working range and extend their life time, to reduce cost, and to improve quality under uncertain and changing circumstances. System-level control loops play an increasingly important role. What is the best way to co-design control and embedded hardware and software?

To achieve industrial acceptance, we need systematic, semi-automatic DSE methods that can cope with the complexity of next generations of high-tech systems. These methods should be able to cope with the many different use cases that a typical embedded platform needs to support, and the trade-offs that need to be made between the many objectives that play a role in DSE. Model versioning and tracking of decision making need to be supported. Model calibration and model validation are other important aspects to take into account.

Model-driven DSE as presented in this chapter aims to support decision making early in the development process. It needs to be connected to other phases in development such as coding and code generation, hardware synthesis, and possibly model-based testing. Industrially mature DSE tools are a prerequisite. DSL support, tool chain customisation, integration with other development tools, and training all need to be taken care of.

In conclusion, the views and results presented in this chapter provide a solid basis for model-driven DSE. The motivation for the work comes from important industrial challenges, which in turn generate interesting scientific challenges. It is this combination of industrial and scientific challenges that makes the work particularly interesting. The scientific challenges point out the need for integration of modelling and analysis techniques, and not necessarily the further development of specialised techniques. A stronger focus on integration would benefit the transfer of academic results to industrial practice.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. AADL. http://www.aadl.info/ (2012). Accessed Oct 2012
2. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking in dense real-time. Inf. Comput. **104**, 2–34 (1993)
3. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theor. Comput. Sci. **138**, 3–34 (1995)
4. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**, 183–235 (1994)
5. Apache Felix. http://felix.apache.org/ (2012). Accessed Oct 2012
6. Balarin, F., Giusto, P., Jurecska, A., Passerone, C., Sentovich, E., Tabbara, B., Chiodo, M., Hsieh, H., Lavagno, L., Sangiovanni-Vincentelli, A., Suzuki, K.: Hardware-Software Co-design of Embedded Systems: The POLIS Approach. Kluwer Academic Publishers, Norwell (1997)
7. Basten, T., Hendriks, M., Somers, L., Trčka, N.: Model-driven design-space exploration for software-intensive embedded systems (extended abstract). In: Jurdzinski, M., Nickovic, D. (eds.) Formal Modeling and Analysis of Timed Systems. Lecture Notes in Computer Science, vol. 7595, pp. 1–6. Springer, Berlin (2012)
8. Basten, T., van Benthum, E., Geilen, M., Hendriks, M., Houben, F., Igna, G., Reckers, F., de Smet, S., Somers, L., Teeselink, E., Trčka, N., Vaandrager, F., Verriet, J., Voorhoeve, M., Yang, Y.: Model-driven design-space exploration for embedded systems: The Octopus toolset. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification, and Validation. Lecture Notes in Computer Science, vol. 6415, pp. 90–105. Springer, Heidelberg (2010). http://dse.esi.nl/. Accessed Oct 2012
9. Basu, A., Bozga, M., Sifakis, J.: Modeling heterogeneous real-time components in BIP. In: Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2006), Pune, pp. 3–12 (2006)
10. Behrmann, G., David, A., Larsen, K.G., Hakansson, J., Pettersson, P., Yi, W., Hendriks, M.: Uppaal 4.0. In: Proceedings of the Third International Conference on the Quantitative Evaluation of Systems (QEST06), Riverside, pp. 125–126 (2006). http://www.uppaal.com/. Accessed Oct 2012
11. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – a platform and programming language independent interface for search algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science, vol. 2632, pp. 494–508. Springer, Berlin (2003). http://www.tik.ee.ethz.ch/pisa/. Accessed Oct 2012
12. Börner, K.: Plug-and-play macroscopes. Commun. ACM **54**, 60–69 (2011)
13. Bulychev, P.E., David, A., Larsen, K.G., Mikučionis, M., Poulsen, D.B., Legay, A., Wang, Z.: UPPAAL-SMC: statistical model checking for priced timed automata. In: Proceedings of the 10th Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL 2012), Tallinn, pp. 1–16 (2012)

14. Cassez, F., Larsen, K.: The impressive power of stopwatches. In: Palamidessi, C. (ed.) CONCUR 2000 – Concurrency Theory. Lecture Notes in Computer Science, vol. 1877, pp. 138–152. Springer, Berlin (2000)
15. CoFluent design, CoFluent studio. http://www.cofluentdesign.com/ (2012). Accessed Oct 2012
16. Cyberinfrastructure shell. http://cishell.org/home.html (2012). Accessed Oct 2012
17. Davare, A., Densmore, D., Meyerowitz, T., Pinto, A., Sangiovanni-Vincentelli, A., Yang, G., Zeng, H., Zhu, Q.: A next-generation design framework for platform-based design. In: Proceedings of the 2007 Design and Verification Conference (DVCon 2007), San Jose (2007)
18. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B., van Vliet, J., Wang, Z.: Statistical model checking for networks of priced timed automata. In: Formal Modeling and Analysis of Timed Systems. Lecture Notes in Computer Science, vol. 6919, pp. 80–96. Springer, Berlin (2011)
19. Derler, P., Lee, E.A., Matic, S.: Simulation and implementation of the PTIDES programming model. In: Proceedings of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '08), Vancouver, pp. 330–333 (2008)
20. Eker, J., Janneck, J.W.: CAL language report specification of the CAL actor language. ERL technical memo UCB/ERL M03/48, University of California, Berkeley (2003)
21. Eker, J., Janneck, J.W., Lee, E.A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y.: Taming heterogeneity – the Ptolemy approach. Proc. IEEE **91**, 127–144 (2003)
22. Esterel technologies, SCADE Suite. http://www.esterel-technologies.com/products/scade-suite (2012). Accessed Oct 2012
23. Hendriks, M., Geilen, M., Basten, T.: Pareto analysis with uncertainty. In: Proceedings of the 9th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2011), Melbourne, pp. 189–196 (2011)
24. Hendriks, M., Vaandrager, F.W.: Reconstructing critical paths from execution traces. In: Proceedings of the 10th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2012), Paphos, pp. 524–531 (2012)
25. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston (2004). http://spinroot.com/. Accessed Oct 2012
26. Houben, F., Igna, G., Vaandrager, F.: Modeling task systems using parameterized partial orders. Int. J. Softw. Tools Technol. Transf. (2012). DOI: 10.1007/s10009-012-0264-8
27. Hsu, C.J., Keceli, F., Ko, M.Y., Shahparnia, S., Bhattacharyya, S.S.: DIF: an interchange format for dataflow-based design tools. In: Pimentel, A.D., Vassiliadis, S. (eds.) Computer Systems: Architectures, Modeling, and Simulation. Lecture Notes in Computer Science, vol. 3133, pp. 3–32. Springer, Berlin (2004)
28. IBM ILOG CPLEX optimizer. http://www.ibm.com/CPLEX/ (2012). Accessed Oct 2012
29. Igna, G., Kannan, V., Yang, Y., Basten, T., Geilen, M., Vaandrager, F., Voorhoeve, M., de Smet, S., Somers, L.: Formal modeling and scheduling of data paths of digital document printers. In: Cassez, F., Jard, C. (eds.) Formal Modeling and Analysis of Timed Systems. Lecture Notes in Computer Science, vol. 5215, pp. 170–187. Springer, Heidelberg (2008)
30. Igna, G., Vaandrager, F.: Verification of printer datapaths using timed automata. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification, and Validation. Lecture Notes in Computer Science, vol. 6416, pp. 412–423. Springer, Heidelberg (2010)
31. Improvise. http://www.cs.ou.edu/~weaver/improvise/index.html (2012). Accessed Oct 2012
32. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. MIT Press, Cambridge (2006)
33. Jackson, E.K., Kang, E., Dahlweid, M., Seifert, D., Santen, T.: Components, platforms and possibilities: towards generic automation for MDA. In: Proceedings of the Tenth ACM International Conference on Embedded Software (EMSOFT 2010), Scottsdale, pp. 39–48 (2010)
34. JAVA genetic algoritms package. http://jgap.sourceforge.net/ (2012). Accessed Oct 2012
35. Jensen, K., Kristensen, L., Wells, L.: Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. Int. J. Softw. Tools Technol. Transf. **9**, 213–254 (2007)

36. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, Berlin (2009)
37. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. In: Proceedings of the Sixth International Conference on the Quantitative Evaluation of Systems (QEST 2009), Budapest, pp. 167–176 (2009)
38. Keinert, J., Streubühr, M., Schlichter, T., Falk, J., Gladigau, J., Haubelt, C., Teich, J., Meredith, M.: SystemCoDesigner–an automatic ESL synthesis approach by design space exploration and behavioral synthesis for streaming applications. ACM Trans. Des. Autom. Electron. Syst. 14(1), Article 1, 1:1–1:23 (2009). http://www12.informatik.uni-erlangen.de/research/scd/. Accessed Oct 2012
39. Kienhuis, B., Deprettere, E., Vissers, K., van der Wolf, P.: An approach for quantitative analysis of application-specific dataflow architectures. In: Proceedings of the 1997 IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP '97), Zurich, pp. 338–349 (1997)
40. Kumar, A.: Adding schedulability analysis to the Octopus toolset. Master's thesis, Eindhoven University of Technology, Faculty of Mathematics and Computer Science, Design and Analysis of Systems group, Eindhoven (2011)
41. Lee, E.A., Messerschmitt, D.G.: Static scheduling of synchronous data flow programs for digital signal processing. IEEE Trans. Comput. **36**, 24–35 (1987)
42. Lukasiewycz, M., Glaß, M., Reimann, F., Teich, J.: Opt4j: meta-heuristic optimization framework for Java. In: Proceedings of the 13th Annual Conference Genetic and Evolutionary Computing Conference (GECCO 2011), Dublin, pp. 1723–1730 (2011). http://opt4j.sourceforge.net/. Accessed Oct 2012
43. MathWorks – global optimization toolbox – Solve multiple maxima, multiple minima, and nonsmooth optimization problems. http://www.mathworks.com/products/global-optimization (2012). Accessed Oct 2012
44. MathWorks – SimEvents – discrete-event simulation software. http://www.mathworks.com/products/simevents/ (2012). Accessed Oct 2012
45. MathWorks – Simulink – simulation and model-based design. http://www.mathworks.com/products/simulink/ (2012). Accessed Oct 2012
46. MLDesign Technologies, MLDesigner. http://www.mldesigner.com/ (2012). Accessed Oct 2012
47. Modelica and the Modelica Association. http://www.modelica.org/ (2012). Accessed Oct 2012
48. Moily, A.: Supporting design-space exploration with synchronous data flow graphs in the Octopus toolset. Master's thesis, Eindhoven University of Technology, Faculty of Mathematics and Computer Science, Software Engineering and Technology group, Eindhoven (2011)
49. Nikolov, H., Thompson, M., Stefanov, T., Pimentel, A., Polstra, S., Bose, R., Zissulescu, C., Deprettere, E.: Daedalus: toward composable multimedia MP-SoC design. In: Proceedings of the 45th Annual Design Automation Conference (DAC 2008), Anaheim, pp. 574–579 (2008). http://daedalus.liacs.nl/. Accessed Oct 2012
50. NuSMV. http://nusmv.fbk.eu/ (2012). Accessed Oct 2012
51. Open SystemC Initiative (OSCI). http://www.systemc.org/ (2012). Accessed Oct 2012
52. OSGi Alliance: OSGI service platform release 4. http://www.osgi.org/Specifications/HomePage (2012). Accessed Oct 2012
53. Pareto, V.: Manual of Political Economy (manuale di economia politica). Kelley, New York (1971 (1906)). Translated by A.S. Schwier, A.N. Page
54. PRISM. http://www.prismmodelchecker.org/ (2012). Accessed Oct 2012
55. ProM – process mining workbench. http://www.promtools.org/prom6/ (2012). Accessed Oct 2012
56. Qt – a cross-platform application and UI framework. http://qt.nokia.com/products/ (2012). Accessed Oct 2012
57. RTCtoolbox: modular performance analysis with real-time calculus. http://www.mpa.ethz.ch/Rtctoolbox/ (2012). Accessed Oct 2012

58. Sander, I., Jantsch, A.: System modeling and transformational design refinement in ForSyDe. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **23**, 17–32 (2004)
59. Schindler, K.: Measurement data visualization and performance visualization. Internship report, Eindhoven University of Technology, Department of Mathematics and Computer Science (2008)
60. Stuijk, S., Geilen, M., Basten, T.: Throughput-buffering trade-off exploration for cyclo-static and synchronous dataflow graphs. IEEE Trans. Comput. **57**, 1331–1345 (2008)
61. Stuijk, S., Geilen, M., Theelen, B., Basten, T.: Scenario-aware dataflow: modeling, analysis and implementation of dynamic applications. In: Proceedings of the 2011 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XI), Samos, pp. 404–411 (2011)
62. Stuijk, S., Geilen, M.C.W., Basten, T.: SDF^3: SDF for free. In: Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD 2006), Turku, Finland, pp. 276–278 (2006)
63. Symtavision SymTA/S. http://www.symtavision.com/symtas.html/ (2012). Accessed Oct 2012
64. SysML. http://www.sysml.org/ (2012). Accessed Oct 2012
65. Teeselink, E., Somers, L., Basten, T., Trčka, N., Hendriks, M.: A visual language for modeling and analyzing printer data path architectures. In: Proceedings of the Industry Track of Software Language Engineering 2011 (ITSLE 2011), Braga, pp. 1–20 (2011)
66. Theelen, B.D., Florescu, O., Geilen, M.C.W., Huang, J., van der Putten, P.H.A., Voeten, J.P.M.: Software/hardware engineering with the parallel object-oriented specification language. In: Proceedings of the 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE '07), Nice, pp. 139–148 (2007)
67. Theelen, B.D., Geilen, M.C.W., Basten, T., Voeten, J.P.M., Gheorghita, S.V., Stuijk, S.: A scenario-aware data flow model for combined long-run average and worst-case performance analysis. In: Proceedings of the Fourth ACM and IEEE International Conference on Formal Methods and Models for CoDesign (MEMOCODE 2006), Napa, pp. 185–194 (2006)
68. TimeDoctor. http://sourceforge.net/projects/timedoctor/ (2012). Accessed Oct 2012
69. Trčka, N., Hendriks, M., Basten, T., Geilen, M., Somers, L.: Integrated model-driven design-space exploration for embedded systems. In: Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XI), Samos, pp. 339–346 (2011). http://dse.esi.nl/. Accessed Oct 2012
70. Trčka, N., Voorhoeve, M., Basten, T.: Parameterized timed partial orders with resources: formal definition and semantics. ES report ESR-2010-01, Eindhoven University of Technology, Department of Electrical Engineering, Electronic Systems group, Eindhoven (2010)
71. Trčka, N., Voorhoeve, M., Basten, T.: Parameterized partial orders for modeling embedded system use cases: formal definition and translation to coloured Petri nets. In: Proceedings of the 11th International Conference on Application of Concurrency to System Design (ACSD 2011), Kanazawa, pp. 13–18 (2011)
72. UML – Object Management Group. http://www.uml.org (2012). Accessed Oct 2012
73. UML profile for MARTE: modeling and analysis of real-time and embedded systems. http://www.omgmarte.org/ (2012). Accessed Oct 2012
74. van Beek, D.A., Collins, P., Nadales, D.E., Rooda, J.E., Schiffelers, R.R.H.: New concepts in the abstract format of the compositional interchange format. In: Proceedings of the 3rd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS 2009), Zaragoza, pp. 250–255 (2009)
75. Viskic, I., Yu, L., Gajski, D.: Design exploration and automatic generation of MPSoC platform TLMs from Kahn Process Network applications. In: Proceedings of the ACM SIGPLAN/SIGBED 2010 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '10), Stockholm, pp. 77–84 (2010)
76. Yang, Y.: Exploring resource/performance trade-offs for streaming applications on embedded multiprocessors. Ph.D. thesis, Eindhoven University of Technology, Eindhoven (2012)
77. Yang, Y., Geilen, M., Basten, T., Stuijk, S., Corporaal, H.: Exploring trade-offs between performance and resource requirements for synchronous dataflow graphs. In: Proceedings

of the 7th IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia 2009), Grenoble, pp. 96–105 (2009)

78. Yang, Y., Geilen, M., Basten, T., Stuijk, S., Corporaal, H.: Automated bottleneck-driven design-space exploration of media processing systems. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2010), Dresden, pp. 1041–1046 (2010)
79. Yang, Y., Geilen, M., Basten, T., Stuijk, S., Corporaal, H.: Playing games with scenario- and resource-aware SDF graphs through policy iteration. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2012), Dresden, pp. 194–199 (2012)
80. Yices: an SMT solver. http://yices.csl.sri.com/ (2012). Accessed Oct 2012

Chapter 8
Engineering Adaptive Embedded Software: Managing Complexity and Evolution

Kardelen Hatun, Arjan de Roo, Lodewijk Bergmans, Christoph Bockisch, and Mehmet Akşit

Abstract Software plays an increasingly important role in the development of electronic systems. In particular, software is used to control the behaviour of systems in advanced ways that enable system features that would not be feasible otherwise. Making such an embedded system adaptive can improve its performance in certain situations, or extend its applicability to a broader range of situations. In this chapter we explain why this is the case, and how adaptivity can provide a competitive advantage. However, realising and maintaining adaptive embedded software brings its own challenges, sometimes even so prohibitive that the benefits of adaptivity are given up. We explain how adaptivity compromises the ability to manage software complexity and the ability to maintain evolving embedded software. To improve on this, we present our approach of a systematic method towards the development of adaptive embedded software. Two case studies explain two concrete applications of this approach: The first application is a method and corresponding tool set for developing flexible (adaptive) schedulers. It is shown how to use this method to develop application-specific schedulers in a modular way, exploiting a domain-specific language for concisely describing schedulers. We demonstrate that evolving requirements can be handled conveniently. The second application is a method that supports the development of Multi-Objective Optimisations, especially for physical control problems. We discuss the software engineering challenges involved in developing such systems, and explain the various steps and domain-specific languages of the method. Then we illustrate how this method was applied to an industrial case.

K. Hatun (✉) • A. de Roo • L. Bergmans • C. Bockisch • M. Akşit
Software Engineering group, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
e-mail: k.hatun@ewi.utwente.nl; a.j.deroo@ewi.utwente.nl; l.m.j.bergmans@ewi.utwente.nl; c.m.bockisch@ewi.utwente.nl; m.aksit@ewi.utwente.nl

8.1 Motivation

As the amount of software involved in electronic systems grows – millions of lines of code are not unusual – managing the complexity of the software is a serious concern to development organisations. Moreover, embedded software is typically long-lived, and thus has to be maintained; such software also usually evolves over time: Customers more and more expect the feature set of software-based systems to grow with new software updates.

Making an embedded system *adaptive* can improve its performance in a broader range of situations. In the next section we will explain why this is the case, and how adaptivity can provide a competitive advantage. In this chapter we will explain what the challenges are in creating maintainable, adaptive embedded software, present the approach we have taken to counter these challenges; and we present two case studies that demonstrate this approach. These case studies have taken place in the context of the development of software for professional printers.

The first case study addresses adaptive behaviour with respect to task scheduling. A scheduler aims to optimise system behaviour by ordering tasks in the best possible way, considering given goals and constraints. They are typically tightly integrated with the rest of the system and therefore difficult to maintain or even replace.

The second case study deals with control of physical (sub-)systems; in particular, control problems in a continuous domain. Optimising control typically involves making trade-offs between multiple optimisation targets, which may even change at run-time. For example, the user decides for certain print jobs to value speed over quality or energy consumption, but for the next job image quality is be considered more important.

8.1.1 Adaptivity Provides a Competitive Advantage

Adaptive control can provide a competitive advantage by enabling a system to achieve better performance with the same hardware characteristics, only through better *control*. Our scope is control problems with multiple parameters, which may vary during the operation of the system. There are several types of parameters; some of those are set to achieve the desirable behaviour, some are a reflection of the context, such as environmental conditions, peer systems, and user requests.

We base our terminology used in the remainder of this chapter on the terms introduced in Sect. 2.3.2. All parameters, i.e. *decision variables* and *dependent variables*, are subject to *system constraints* for proper operation of the system: For example, in a printer, the speed of paper transportation must be limited, because at too high speeds, paper handling will fail. But also, the speed may be constrained, depending on parameters like the degree of paper heating, and the type of image being produced.

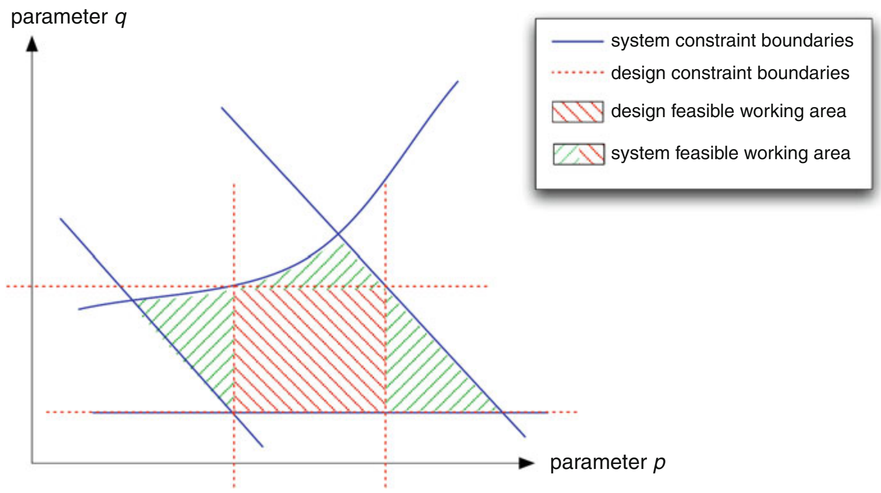

parameter q
parameter p
system constraint boundaries
design constraint boundaries
design feasible working area
system feasible working area

Fig. 8.1 Illustration of the constrained working area for two parameters

Figure 8.1 illustrates the space of possible system parameters through the example of a control problem with two parameters, *p* and *q*, which may vary independently. The solid lines designate the upper and lower boundaries of system constraints. To avoid that the system will operate outside the valid boundaries, typically the involved parameters are constrained at design-time to values where this can be guaranteed: This is shown by the dashed *design constraint* boundaries in the figure. These define a rectangular design working area where all constraints (both design constraints and system constraints) are guaranteed to be met.

Figure 8.2 shows that with the same system constraint boundaries, it is possible to define another design working area, where the parameters get different design constraints. In this case the viable range for parameter *q* is extended (allowing it to reach higher values), thereby substantially reducing the viable range for parameter *p*. This alternative working area can be defined either as a design-time engineering decision, which defines the characteristics of the particular system. Alternatively, systems can be operated in other *modes*, where each mode defines its own designed working area. This is a common practise, that can have a negative impact on the control software: Assume the control software is designed such that at many decision points in the software, the current mode will influence the behaviour. Then in all these locations it has to be checked first in which mode the system is operating, and different control actions are executed depending on the result. It is therefore important (and there are viable techniques for this, often based on explicit state machine models) to pay explicit attention to structure such behaviour; otherwise the software becomes easily very complex, error-prone, and difficult to understand and maintain.

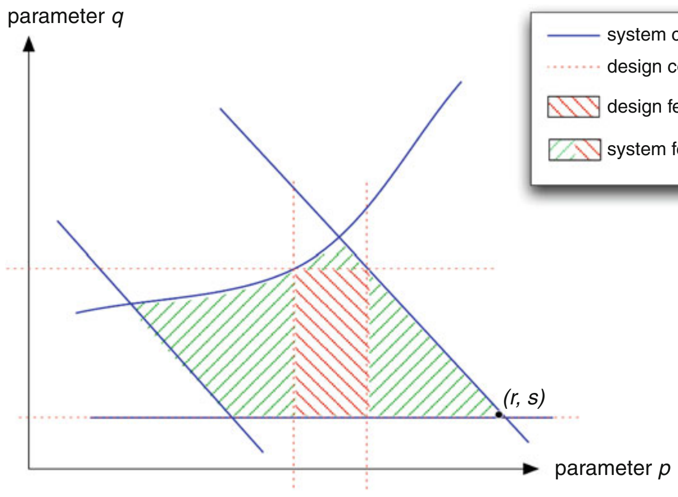

parameter q
parameter p
(r, s)
system constraint boundaries
design constraint boundaries
design feasible working area
system feasible working area

Fig. 8.2 Different design-time trade-off of the design working area

However, even when working with multiple modes or viable ranges, this still leaves parts of the system's feasible working area unexploited, because they cannot be reached while guaranteeing all design constraints are satisfied. Consider for example the point (r,s) in Fig. 8.2: this point lies within the system constraint boundaries, but it can only be reached by relaxing the design constraints on parameter p. Parameter q then needs to be controlled carefully, as it can no longer vary arbitrarily, even within its design constraints. This does require making a *trade-off* between parameters p and q; here making the trade-off to keep the value of parameter q low, thereby enabling a higher value for parameter p. Note that it is possible that some parameters cannot be controlled, but are *dependent variables*.

As a result, the viable operational range of the system has been extended; much higher values for p can now be reached; if p is, e.g. productivity, or quality, or some other feature for which high values are attractive, a more competitive system (with "better" specifications) has been obtained. To be clear: the system has only become "better" by allowing it to make trade-offs , such as delivering higher quality at lower speeds, whereas without adaptivity, the engineering efforts would be focused, e.g. on improving the quality without sacrificing speed or other qualities.

Finally, we would like to point out that the above situations describe snapshots during the development life-cycle and life-time of a system: as a system is refined and improved, the situation will inevitably change. Accordingly, system constraints may change, or new system constraints may be added. Figure 8.3 illustrates this, sketching entirely different system working areas and designed working areas.

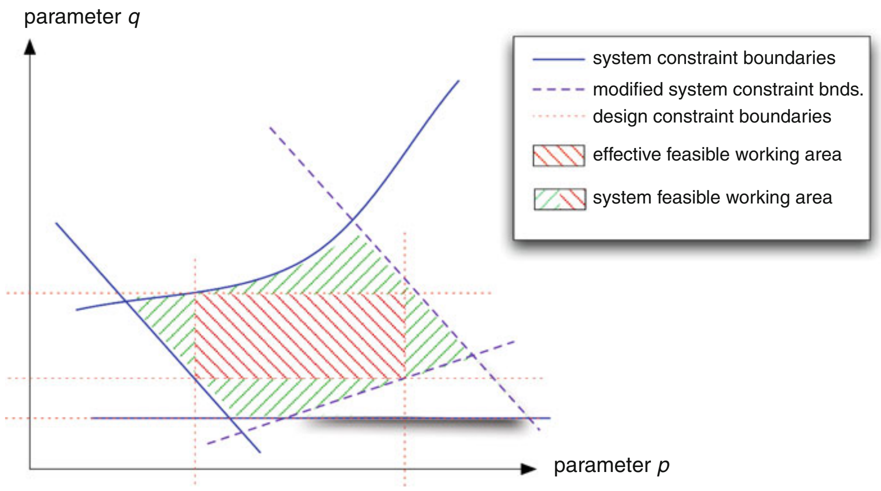


Fig. 8.3 Evolution of the constrained working area

8.1.2 The Implementation of Adaptivity Affects Software Quality

It is commonly known that there are virtually no limitations to the kinds of behaviour that can be implemented in software: Any behaviour that can be precisely specified, in a mathematical formula, or as an algorithm, can be implemented. The only *theoretical* limitations are the required resources, such as computing power, available memory, or available time.

However, there are pertinent *practical* limitations to the ability to realise and maintain software, in particular for large and complex systems. The major challenges of software development are:

- *Managing complexity*. How can we decompose the software into manageable parts, which can be developed and understood independently? How can we keep the number and complexity of dependencies between those parts manageable? And how can we ensure that those parts can be put together such that the resulting whole works correctly?
- *Managing evolution*. Typically, 75–90% of software development effort and costs are spent *after* the initial delivery of a system; hence the ability to enhance, extend, revise, and correct the behaviour of a software system is crucial.

The answers of software engineering to these challenges are manifold [28, 29], but these are the typical qualities that are strived for:

- *Modularity* to achieve locality of change,
- *Comprehensibility* to be able to understand the behaviour of the software, and

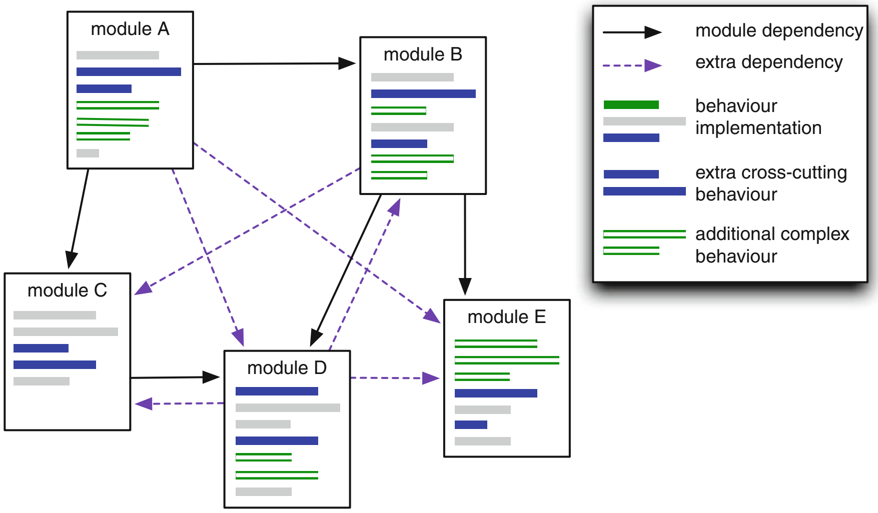


Fig. 8.4 Illustration of the impact of (I.) additional dependencies, (II.) additional cross-cutting behaviour and (III.) more complex behaviour on the structure of the software

- *Evolvability* to be able to modify behaviour without invasively manipulating the previous (and potentially still relevant) version.

We will now discuss three related reasons (also illustrated in Fig. 8.4) why the realisation of adaptivity in software conflicts with these quality goals.

I. Adaptive Behaviour Causes Increased Dependencies between Modules

Software modules aim at addressing particular concerns, such as managing a specific part of the hardware (motors, heaters, inkjet printheads, etc.), implementing different functions (paper transport, paper heating, stapling, etc.), or implementing specific applications (such as various modes of printing, scanning, and copying). A key factor in the manageability of software is the number (and type) of dependencies between modules.

Adaptive control and dynamic optimisation techniques work by carefully monitoring the system state, and accordingly setting or adjusting certain parameters, or selecting appropriate policies. Typically, to make more effective optimisations, more information about the context and state of the system must be taken into account. Similarly, more effective optimisations may adjust a larger set of parameters (of more actuators). As a result, more precise or effective control and optimisations are achieved usually at the cost of causing more dependencies between software modules:

- More sensors and actuators are involved in controlling certain behaviour, including those that are in totally different subsystems (represented by other modules).
- More information about the state of the (software) system is acquired, also from other software modules.
- Information about the activities and plans of other modules may be required to decide upon the best possible settings or optimisation policies.

Note that there can also be *other* reasons for many additional dependencies, such as bad decomposition choices.

II. Adaptive Behaviour is Cross-Cutting

Besides the fact that adaptive controllers have to interface with a lot of modules, the adaptive behaviour itself is typically *scattered* (either distributed, or replicated) over multiple modules; examples are:

- Gathering information from multiple locations (sensor data, state information, activities and plans), or
- Adjusting the values, policies, plans, or activities of multiple other modules, so that these behave in accordance with the desired (adaptive) behaviour.

The implementation of such scattered behaviour is typically tightly interwoven (or *tangled*) with the implementation of the other modules; behaviour that is both scattered and tangled, is also referred to as *cross-cutting behaviour* [14]. Examples involve monitoring the energy usage of the various parts of a system, or coordinated control of multiple printer units (paper feeder, printer, stapling units, sorters, etc.). A result of cross-cutting, adaptive behaviour is that its implementation and maintenance involves adjusting the implementation of multiple modules, which can be painful, but especially bears risks to the stability and reliability of the system.

III. Implementing Adaptivity Results in Higher Complexity

The most simple special case of control is to set parameters to constant values – i.e. to actually not adaptively control–, which clearly has a low complexity. In more powerful controllers, optimisation algorithms set values dynamically, depending on the actual context. Implementing behaviour that is different in many different circumstances, depending on state information, can cause a large increase in the complexity of the code (for example due to many conditionals and control statements). A direct result of the increased dependencies between modules, as well as of scattered and tangled behaviour (as explained under I. and II.) is a much higher complexity of the structure of the software. This will generally result in software of reduced quality, which is certainly more difficult to manage and maintain, resulting in a larger time-to-market and higher development and maintenance costs.

8.1.3 Overview of Evolvability in Embedded Software

A number of modelling approaches have been introduced to manage the complexity of embedded control software [17, 30]. The common feature among them is that they support expressing reactive behaviour using state transition diagrams (STD). The work on UML [25] and ObjecTime/Rose RealTime [27] has focused more on how to structure STD-based specifications within the context of an object-oriented design.

A key feature of the above approaches is that they lead to implementations whose structure is dominated by the STDs that specify the reactive behaviour. Either the program as a whole is dominated by this STD structure, or at least the implementation of some modules within the program is dominated by it. This is clearly suitable for those parts of the system where that reactive behaviour is actually dominant, but this is not always the case: Embedded system design involves many more concerns, including physical (continuous-time) models, optimisation problems, activity-scheduling issues, resource management, and many more.

Essentially, STDs are a notation, or language, specifically designed for the domain of event-based, reactive behaviour. We will argue that, next to STD-based development methods, alternative domain-specific approaches are needed: They allow the development of solutions in their respective domains such that they retain the modularity of the problem domain and hence become much easier to develop, maintain, and evolve. One of the key challenges then is to combine these different modelling techniques into well-integrated systems.

It is a key issue of the approach presented in this chapter to separate specific parts of the system, while being able to compose these modules into an integrated system. This compositional power distinguishes our approach from the majority of other work in the area of Domain-Specific Languages (DSLs) [24], Model-Driven Engineering [21], and Generative Programming [7].

8.2 Approach

In the remainder of this chapter we will present a systematic approach of the development of adaptive software. The aim of this approach is to *enable the – efficient – development of adaptive software without sacrificing the quality of the software system.* In this section we present the general approach, the subsequent sections show two concrete cases that demonstrate the application of the approach to two different domains:

- Section 8.3 discusses the development of flexible task scheduling. This section extends an earlier publication of our approach for developing a scheduler workbench [18].
- Section 8.4 presents our method, called the MO2 method, for the development of (adaptive) multi-objective optimisation of system behaviour. This method is

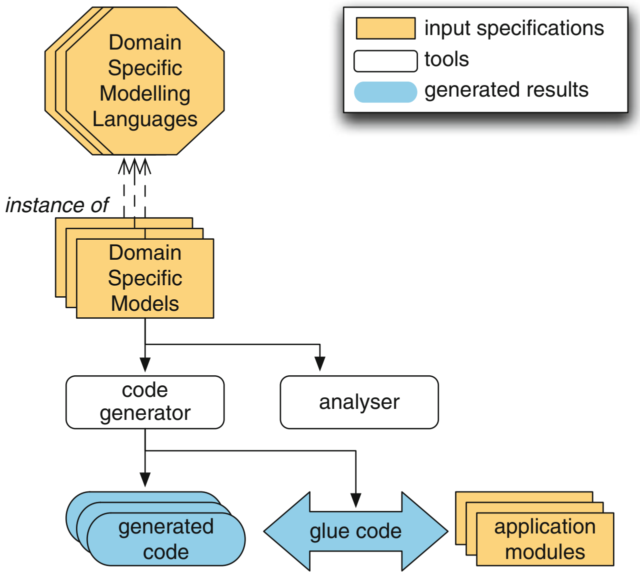


Fig. 8.5 The core ingredients of our approach towards designing adaptive software

illustrated by a case study. More details about this method and the case study can be found in [11, Chaps. 4 and 5] and [10], which improves over an earlier version [9].

Our approach is based on the model-driven paradigm; we propose to develop, or adopt, models that focus on the domain that needs to be made adaptive, for example the task scheduling domain, or the physical domain. These are so-called *Domain-Specific Models* (DSMs). We have several motivations for this:

1. Domain-specific models are closer to the concepts and way of thinking of domain experts, and are usually more concise and expressive with respect to the domain: This makes it easier to manage their complexity.
2. Developing models for a system's adaptive behaviour causes a separation of the domain-specific elements from other concerns; this creates a modularity in the model that can reduce its complexity and improve maintainability (among others as a result of locality of changes).
3. One of the goals of models is to focus on expressing what needs to be done, rather than how; the latter can then be expressed separately, as part of the process of going towards an implementation.

Domain-specific models need to be expressed in a domain-specific (modelling) language (DSL). If for a specific domain no suitable modelling languages are available, they will need to be defined first.

Figure 8.5 provides an overview of the approach, and shows that one or more domain-specific models, expressed as instances of domain-specific modelling languages, serve as input to some tools: One of the tools is an *analyser*, which exploits the knowledge about the domain to analyse the models, e.g. for correctness constraints, feasibility, and performance criteria. Another tool is a *code generator*, which takes the domain-specific models as input, and uses those, together with knowledge about the domain, such as specific techniques, algorithms, and solutions, to generate code that implements the domain-specific models.

The domain-specific models normally describe only a part of the system, whereas the remainder is implemented by *application modules*, which are typically implemented in a General Purpose Language (GPL), such as Java, C#, C++, etc. Hence application modules must be integrated (or at least they must interface) with the generated code from the domain-specific models; similarly, the code from multiple domain-specific models must be integrated: This is the purpose of the glue code, which is partially created by the code generator and partially hand-crafted to match the target system. As a result, the application modules, as well as the domain-specific models, remain independent of each other, regardless of how they are integrated, and the various modules can evolve mostly independently; all code that is needed to integrate the generated component with the rest of the system is localised in the glue code.

The above description of our approach omits many details; a more complete picture is provided in Fig. 8.6. In this figure yellow clouds represent knowledge that is involved in the process explicitly: To define an appropriate domain-specific modelling language, ample knowledge and understanding of the specific domain is required, as indicated by the arrow from "domain knowledge" to "Domain-Specific Modelling Languages". Each domain-specific model is really an application-specific model, expressed in a domain-specific modelling language. Creating a domain-specific model hence involves both general knowledge of the domain, as well as application-specific knowledge. The application modules are also based on *application* domain knowledge. A special part of the domain knowledge is formed by algorithms and solution techniques from the domain, knowledge of these is used to design the code generator and analyser tools.

A particular characteristic of composing domain-specific models and application modules is that the first tends to inject dependencies into all other modules in the system. Thus, in a direct implementation the code generated from the domain-specific models would be cross-cutting the application modules. To avoid this, the glue code is responsible for establishing the required links, while retaining the modularity property of all generated and application code. This balancing act can be achieved with appropriate tools, which can "weave" for example generated domain-specific code into application code. Weaving is a technology adopted from aspect-oriented programming (AOP) [15]. A *code weaver* tool can operate on the generated code, and does not need to change the source code. Thus, code generated from modules remains separate and modular with respect to application modules, so that the latter can still be properly maintained.

Figure 8.6 explicitly shows the generated executable application, and the output of the analyser (typically reports for the model designer) as distinct documents (the green, rounded rectangles).

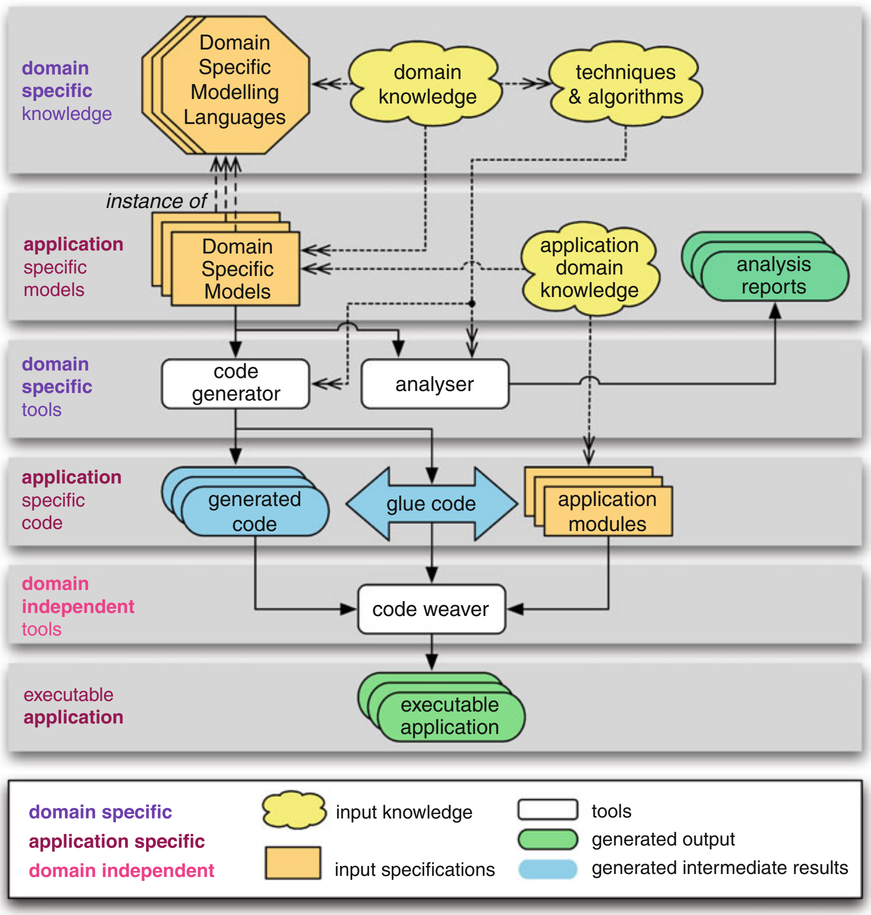


Fig. 8.6 An overview of our model-based approach of designing adaptive systems

We can distinguish several layers in the approach (the grey boxes), each with its distinct purpose and abstraction level; at the top the domain-specific (but application-independent) knowledge is located. This layer serves as input for constructing the domain-specific models and the code generator and analyser tools. The latter two are *domain*-specific tools, but not *application*-specific. The code they generate and the application modules, however, *are* application-specific, as well as the executable application that is the output of the code weaver. A code weaver itself is a generic tool that is independent of any application domain.

In the following sections we will illustrate this general approach with two concrete case studies we have performed within the field of professional printing. Examples of involved domains are physics (thermodynamics, etc.) and planning of tasks for finishing a print job.

8.3 Flexible Task Scheduling with an Automatically Generated Scheduler

One specific domain in which embedded controller software frequently performs optimisation is that of *allocating resources to jobs over a period of time while optimising one or more objectives* [4]. The component performing this optimisation is called a *scheduler*; it has a direct impact on resource management and, as a consequence, system performance. It is evident that a scheduler communicates with multiple system components, therefore its implementation is likely to be highly coupled. The design issues attached to a system scheduler can be categorised in two parts:

1. *Specification of scheduler components.* A scheduler component consists of (1) information about the base system in the form of data structures, (2) communication interfaces with the base system, and (3) a *scheduling algorithm* that can solve the scheduling problem according to the objectives of that base system. The concrete definition of these elements may vary greatly depending on the application area: The scheduling requirements (on information, communication, and objective) of a safety-critical system are very different from those of an operating system. The challenge lies in expressing system characteristics and the desired scheduling algorithm at the same level and defining the scheduling algorithm with the tasks and resources of the base system.
2. *Non-intrusive integration of schedulers.* Recalling the scheduling definition in the first paragraph, it is obvious that a scheduler has to constantly communicate about the current state of the resources and tasks, and at the same time must enforce constraints which may be predefined or introduced on-the-fly. These aspects alone pose a challenge in creating expressive and flexible interfaces for the scheduler. However there is another challenge which is the by-product of software/system evolution. Since schedulers involve system-specific parts, when systems evolve it is necessary to alter or – depending on the amount of change performed in the system – completely replace the scheduler. This is difficult given the scheduler's tight integration within the system software. This problem calls for a non-intrusive integration mechanism between the scheduler and the system and a cost-effective way of altering/replacing the scheduler implementation.

8.3.1 Scheduling Workbench

We have developed a *Scheduling Workbench*, to *model and integrate schedulers into existing applications* (an overview is depicted in Fig. 8.7). We have *not* developed new scheduling algorithms, but provide a means to easily integrate schedulers into a system. Our "domain-specific scheduling language" acts as a meta-model and the entities involved in the scheduling process as well as the interface between them

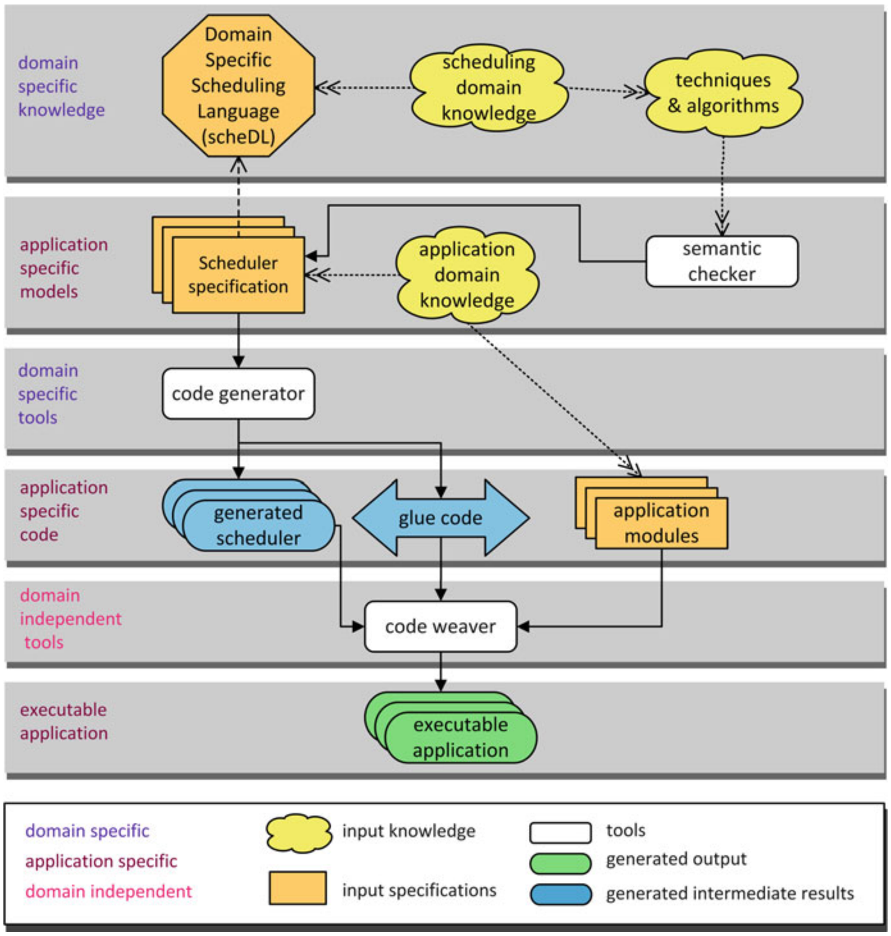


Fig. 8.7 Our approach instantiated for the scheduling domain

and a scheduling algorithm can be defined in this meta-model. Thus, "generated scheduler" refers to code which is generated from the "scheduler specification" and which interfaces with an implementation of a scheduling algorithm.

The first step of development of such a workbench is a comprehensive domain analysis. Domain analysis is the process where the fundamental domain concepts and their variability are determined. In domain modelling these concepts and their relationships are expressed in a model. In Fig. 8.7 this model is shown as *scheduling domain knowledge*. From this model we have derived a grammar for a scheduling domain-specific language, shown as an octagon in Fig. 8.7 connected to scheduling domain knowledge.

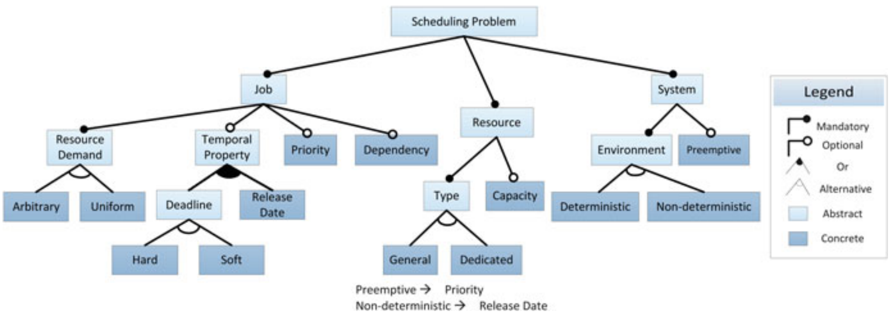


Fig. 8.8 A simplified version of the feature model for scheduling

8.3.1.1 Domain Analysis and Modelling

For domain analysis we have analysed commonalities and variabilities of scheduling approaches. We started by extracting the core domain concepts by means of a literature study and identifying the building blocks of the scheduling domain. After finding the core concepts, we have analysed the variations of these concepts. We have expressed our findings about the domain using feature modelling [19].

In Fig. 8.8 a feature model for scheduling is shown (the full model can be found in [18]). In the feature model notation every domain concept is mapped onto abstract features and their variations are mapped onto concrete features. For example, if we look at the **deadline** feature, we see that a deadline can be either **hard** or **soft**. It is also possible to define cross-tree constraints as logical expressions, shown in Fig. 8.8 below the feature tree. This is additional semantic information about the domain which is used to implement the *semantic checker* shown in Fig. 8.7.

8.3.1.2 Domain-Specific Language Design and Code Generation

Following the advice of Mernik et al. [24], we have designed a DSL for the scheduling domain (the "scheduling domain-specific language", scheDL) using the feature model presented in the previous subsection. We have used the feature model as the abstract syntax and turned it into the concrete grammar of our language, especially by defining keywords. In our approach we chose to generate *general-purpose language (GPL)* code from our DSL, which is called generative programming [7]. This method allows a user to program on a higher abstraction layer, and to obtain code in widely supported and robust programming languages (e.g. Java, or C++) at the same time. The *code generator* knows how to process a *scheduler specification* and how to incorporate it with *application domain knowledge* to obtain GPL code (cf. the arrow from "application domain knowledge" to "code generator" in Fig. 8.7).

Let us illustrate the benefits of our approach by explaining how we handle the two main challenges presented in Sect. 8.3. The first one expresses the requirements of a scheduling problem using the concepts found in the scheduling domain. At the end of the domain analysis and modelling phase, we have a model of the scheduling domain, which includes core concepts, their variations and the relationships of those concepts. From this model we have created a grammar for scheDL. Using this language we can define jobs, resources, and schedulers with concise domain-specific language constructs.

The block structure syntax of scheDL promotes readability and makes it accessible even to non-programmers. Using a DSL we are able to describe the scheduling requirements and a scheduler concisely. This is also useful for maintainability since altering the scheduler only requires changing the short specification file written in scheDL.

The second challenge mentioned in Sect. 8.3 concerns a scheduler's tight integration with a system. In scheDL we offer language constructs to model behavioural interactions between the system and the scheduler in the form of event declarations. During code generation these event declarations are turned into software components of a special kind, called *aspects*. The language mechanism of aspects enables to compose the behaviour of aspect components with the events of other components without intrusively modifying them to make the events explicit in the code.

8.3.2 Example Cases

We have tested our Scheduling Workbench on the `DemoPrinter` application, which has been developed in Java and simulates a professional printer. Figure 8.9 shows the static structure of this application; for brevity we have left out some utility classes from this view. In the following, we demonstrate out workbench by applying to three example cases. The examples are written in scheDL[1]; the examples are (1) replacing the existing scheduler of the demo printer, (2) adding a new hardware component to be scheduled, and (3) supporting multiple scheduling policies with the new scheduler.

8.3.2.1 Replacing the Scheduler in Legacy Code

In this example case, we replace the scheduler of the `DemoPrinter` using the Scheduling Workbench. The first step is to write a scheDL specification describing the kind of jobs and resources exist in the system. We also define a scheduler in the specification. In Listing 8.1 the definition of a **`Job`** is shown. A **`Job`** with the

[1]In listings, scheDL language keywords are shown in bold.

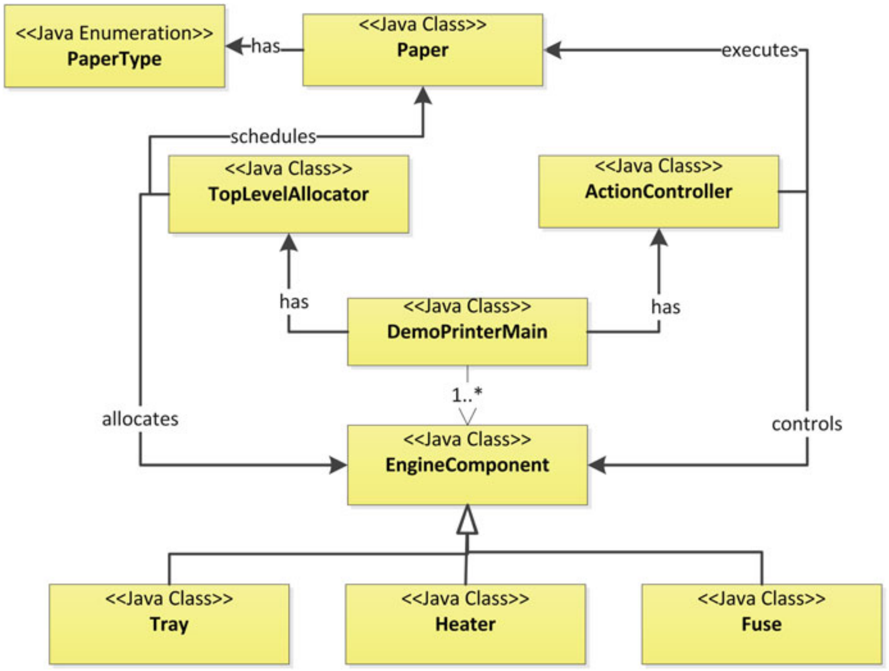


Fig. 8.9 The static structure of `DemoPrinter`

name `Paper` is defined. This job has an integer, the so-called release date, which is defined as the time a job becomes available. It is also possible to define properties like deadline, or priority. In the definition we also reference **`Executor`** resources as a **`DEMAND`**, which means: In order to be able to execute a `Paper` job, we need a `Tray`, a `Heater`, and a `Fuse`. The `newpath` property is a *path declaration* which defines an order between the demanded resources. The scheduler has to consider processing times of the involved component such that paper jams are avoided and it can be ensured that components are ready when the paper arrives.

Listing 8.1 Job definition.
```
Job Paper{
    RELEASE release_date Integer
    DEMAND Tray
    DEMAND Heater
    DEMAND Fuse
    newpath
}

PathDeclaration newpath{ Tray->Heater->Fuse }
```

In Listing 8.2 the definition of an **`Executor`** is shown. Every structure defined in scheDL must have a unique name. Here we have defined the executor `Tray` as an executing resource with single access, which means it can only execute one job at a time. The same definition is made for `Heater` and `Fuse`.

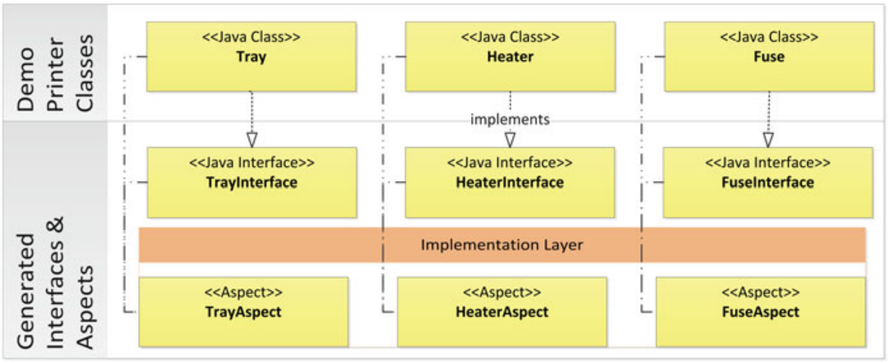


Fig. 8.10 Mapping of system classes and generated interfaces

Listing 8.2 Executor definition.
```
Executor Tray {
    ACCESS single
}
```

The next step is to define the scheduler which will include references to the structures defined before. `Scheduler` is the central structure in scheDL; this is, only the structures referenced by the scheduler will be generated. Therefore, it is possible to define multiple jobs or resources in the same file even if they are not used. In Listing 8.3 the scheduler definition for our example is shown. This scheduler definition basically describes a scheduler which assigns the three resources to sheets applying the *First-Come First-Served* policy.

Listing 8.3 Scheduler definition.
```
Scheduler Myscheduler{
    PolicyDefinitions{
        builtin FCFS
    }
    JobReference{
        Paper
    }
    ResReference{
        Tray
        Heater
        Fuse
    }
}
```

This is all the code needed for defining a scheduler for the `DemoPrinter` example. The jobs and resources are automatically mapped to classes with the same name in the `DemoPrinter` application. Figure 8.10 illustrates this mapping for the engine components of the printer. Printer classes are extended by generated interfaces, but this relationship is encapsulated in aspects (the implementation layer shown in the figure). This way we can add behaviour to the `DemoPrinter` without altering its implementation.

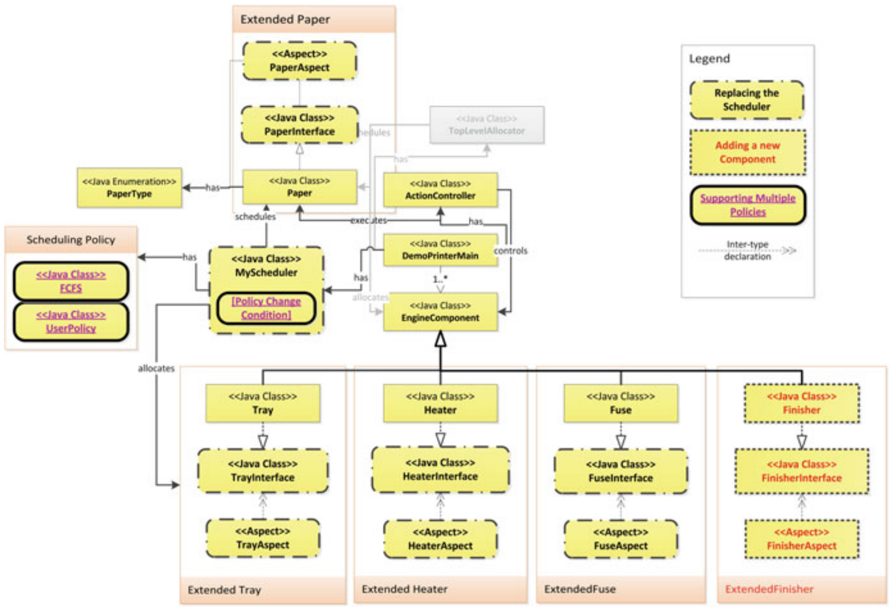

Extended Paper
<<Aspect>> PaperAspect
<<Java Class>> PaperInterface
<<Java Class>> Paper
<<Java Enumeration>> PaperType
<<Java Class>> TopLevelAllocator
<<Java Class>> ActionController
<<Java Class>> DemoPrinterMain
<<Java Class>> EngineComponent
<<Java Class>> MyScheduler
[Policy Change Condition]
Scheduling Policy
<<Java Class>> FCFS
<<Java Class>> UserPolicy
has
schedules
executes
controls
allocates
1..*
<<Java Class>> Tray
<<Java Class>> TrayInterface
<<Aspect>> TrayAspect
Extended Tray
<<Java Class>> Heater
<<Java Class>> HeaterInterface
<<Aspect>> HeaterAspect
Extended Heater
<<Java Class>> Fuse
<<Java Class>> FuseInterface
<<Aspect>> FuseAspect
ExtendedFuse
<<Java Class>> Finisher
<<Java Class>> FinisherInterface
<<Aspect>> FinisherAspect
ExtendedFinisher
Legend
Replacing the Scheduler
Adding a new Component
Supporting Multiple Policies
Inter-type declaration

Fig. 8.11 The final view of the system after all example cases

Given a scheDL specification, the Scheduling Workbench outputs code consisting of the user-defined components, the scheduler and its helper classes, a library of scheduling policies declared in the scheDL specification, and abstract classes that capture domain concepts like job, resource, etc.

Since the `DemoPrinter` application already contains a scheduler, the user has to find and disable its code after the generated code is imported, e.g. by removing the `TopLevelAllocator` class in the example. Then the new scheduler can be added to the base system, ready to work. A concise and fixed interface is provided with the scheduler to make it intuitive to use in the base system. The system's state after making this change can be seen in Fig. 8.11. If a scheduler in an application is developed in the Scheduling Workbench in the first place, switching the scheduler becomes even easier: Only the scheDL specification has to be replaced.

8.3.2.2 Adding a Component

In the second example we will illustrate what happens if a new component, namely a `Finisher`, is added to the system. In order to modify our scheduler software we only need to add a few lines of code to our scheDL specification. The applied changes are shown in Listing 8.4.

Listing 8.4 Extra code for adding a component.
```
Scheduler Myscheduler{
Job Paper{
    ...
    DEMAND Finisher
}
Executor Finisher{
    ACCESS single
}
PathDeclaration newpath{
    Tray->Heater->Fuse->Finisher
}
Scheduler Myscheduler{ ...
    ResReference{ ...
        Finisher
    }
}
```

We only had to add five lines to our scheDL specification to make sure that a new class for `Finisher` and the necessary dependencies are generated connecting this class to the scheduler. After altering the specification the scheduler code needs to be regenerated and imported into the system code. Then the system is able to use the new component. The system's state after making this change can be seen in Fig. 8.11.

8.3.2.3 Supporting Multiple Policies

In the last example case, we add support to `MyScheduler` for supporting multiple policies. This means, the base system can switch policies when a condition changes, for example if the power is low it may choose a power-aware policy. This is important since it enhances system adaptivity greatly. In Listing 8.5 we show what must be added to our scheDL specification to support multiple policies. Firstly we define a resource called `Power` with limited capacity.

Listing 8.5 Extra code for supporting multiple policies.
```
Resource Power{
    CAPACITY limited
}
Scheduler Myscheduler{
    PolicyDefinitions{
        builtin FCFS
        new UserPolicy
        Condition Policychange{
        resource: Power
            FCFS = "Power.getCapacity() >= 100";
            Userpolicy = "Power.getCapacity() < 100";
        }
    }
    ...
}
```

Secondly, in the PolicyDefinitions block we declare two policies, one is First-Come First-Served, provided by the Scheduling Workbench's policy library, hence we distinguish it with the keyword `builtin`. The second one is a *new* policy, meaning it will be implemented/provided by the user. If more than one scheduling policy is declared then a *condition* for policy change needs to be defined.

We can define a condition in scheDL with the `Condition` keyword, then referencing the structure triggering the policy change we define when each policy is valid. According to the condition defined in Listing 8.5 the FCFS policy will be applied when the available power is more than or equal to 100 W, and UserPolicy will be applied when available power is less than 100 W. Figure 8.11 illustrates the system after applying this change.

8.3.3 Conclusion

In these example cases we have demonstrated how to create a scheduler with scheDL. We have also discussed the possibility to use custom scheduling policies allowing the user to customise certain aspects of their specification offering enhanced expressiveness. The complete list of benefits provided by Scheduling Workbench are:

Benefit 1. Domain concerns can be expressed intuitively from a higher level of abstraction.
Benefit 2. Complex domain constraints can be defined and checked; modelling errors are reduced.
Benefit 3. Software maintainability is increased through concise and understandable DSL code.
Benefit 4. Software evolution is eased by the underlying rich domain model.
Benefit 5. Integrated editor and IDE support make programming in DSLs easy.
Benefit 6. Code generation highly reduces manual labour spent on customising general-purpose tools.
Benefit 7. Easy system integration of generated code is provided through non-invasive software engineering techniques.

8.4 Multi-Objective Optimisation of System Qualities in Embedded Control Software

As discussed in Sect. 8.1, to optimise the system behaviour with respect to system qualities, the right values for certain parameters have to be selected. Such controllable parameters are often also called *decision variables*. Examples of decision variables in high-end printing systems are the paper transportation speed of the system and the temperature setpoint of a heating device. Optimising multiple system qualities (or system objectives) by influencing a set of decision variables within the boundaries of the system constraints is known as multi-objective optimisation (MOO) [20].

In current design practise, the control logic implemented in *embedded control software* is decomposed into many different controllers, each controlling a part of the system. Such a decomposition makes the control logic easier to comprehend and maintain, as opposed to, for example, a single (black-box) controller that controls all variables. As the software decomposition usually follows the control decomposition, the different controllers are implemented in several software modules in the embedded control software. Certain system qualities, such as power consumption and productivity, are not controlled by a specific controller, but emerge from the behaviour of the system as a whole. So, if we want to influence these system qualities, we have to manipulate and coordinate many controllers, scattered through the embedded control software. This manipulation and coordination of controllers introduces additional structural complexity within the embedded control software.

To the best of our knowledge, there is a lack of systematic methods to design and implement multi-objective optimisation of system qualities in embedded control software. We have observed that, in practise, state-of-the-art attempts to realise multi-objective optimisation leads to:

- Solutions that are tailored to the specific characteristics of the embedded system, and therefore are inflexible in case the physical system changes or evolves.
- Solutions that are tightly integrated into and coupled with the control software modules, making the embedded control software difficult to comprehend and hard to maintain.
- Solutions that are sufficient but sub-optimal, because the implementation of stronger optimisations would be too complex.

As such, the lack of systematic methods to design and implement multi-objective optimisation in embedded control software leads to higher development and maintenance costs [2]. It may also prevent the implementation of multi-objective optimisation all together, if the performance improvement does not outweigh the reduction in software quality and the increase in development and maintenance costs.

In this section we present the *MO2 method*, a systematic method to design and include multi-objective optimisation and dynamic trade-off making in embedded control software. The MO2 method includes an architectural style to specify and document a multi-objective optimisation solution within the architecture of the embedded control software. The architectural style is supported by techniques, implemented in a tool chain to (1) validate the consistency of the solution, (2) include general optimisation algorithms, and (3) generate code that implements the optimisation algorithm and coordination of the control modules.

8.4.1 MO2 Method Overview

Multi-objective optimisation algorithms try to optimise a given set of utility functions (i.e. objectives) for a given set of decision variables. This type of problem was introduced in works on decision making in economy by Edgeworth [12] in the late nineteenth century. Pareto extended the work with the concept of Pareto optimality [26].

Definition 1 (Multi-objective optimisation problem [6]). A multi-objective optimisation problem exists of the following elements:

- A set of decision variables that can be influenced.
- A set of constraints on the decision variables.
- A set of objectives, expressed as functions of the decision variables.

The solution of a multi-objective optimisation problem is the valuation of the decision variables that satisfies the constraints and provides the (Pareto) optimal value for the objective functions (i.e. there is no other possible valuation for the decision variables that satisfies the constraints and provides a better value for the objective functions).

Note that, if there is more than one utility function, there may not be a single optimal solution for this problem. Instead, the solution is a set of Pareto-optimal points [26].

We developed the MO2 method to design and implement control software with multi-objective optimisation. Figure 8.12 shows an overview of the MO2 method. The MO2 method applies two DSLs: the *MO2 architectural style* and the *SIDOPS+ language* for physical models.

Step 1: The MO2 Architectural Style

The *MO2 architectural style* – i.e. a notation for architectural design as well as a methodology how to structure the architecture – (abbreviated as *MO2 style*) supports architecting control software. We call an architectural model created with the MO2 style an *MO2 architectural model* or *MO2 model*. MO2 models are specialisations of Component-and-Connector models [5]. Besides the different control components and their interfaces, MO2 models include the elements input/output variables, decision variables, constraints, and objective functions. As such, the MO2 style supports the design and documentation of embedded control software that includes multi-objective optimisation functionality.

An MO2 model specifies the architecture of the control software, which consists of software components that implement control logic (i.e. *control components*). The interfaces of these software components consist of input and output variables. The control components are composed into a control architecture by connecting output variables to input variables. Furthermore, an MO2 model specifies which

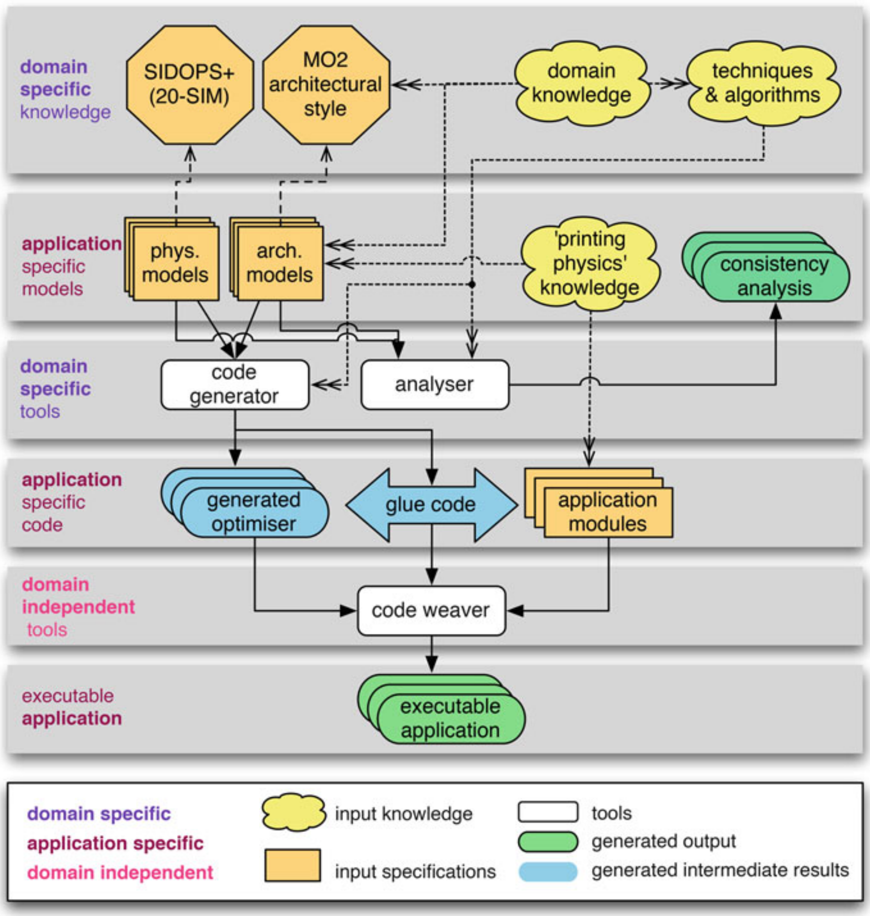


Fig. 8.12 Overview of the MO2 method

variables are decision variables, which variables have constraints (*constrained variables*), and which variables represent the outcome of objective functions (*objective variables*). An MO2 model serves two purposes:

1. It is design documentation of the embedded control software applying MOO.
2. It is an input model to generate an optimiser.

Step 2: Application of the SIDOPS+ Language

The computational logic that is implemented in the software components creates a mathematical relationship between decision variables and input/output variables used in constraints and objective functions. To be able to analyse a MO2 model and generate an optimiser module, the mathematical relationship between the decision variables and the other variables should be specified. Therefore, the MO2 method provides the possibility to refer to models that specify the computational logic (e.g. control logic, or implemented physical characteristics) of components in the architecture. These models can be specified in any language in which computational logic can be mathematically specified. In our implementation of the MO2 method, we apply the SIDOPS+ language of the 20-sim tool set [1, 3, 22], because of the suitability of this language to model control logic and physical characteristics, which are two common domains of computational logic in embedded control software.

Step 3: Analysis and Code Generation

The MO2 method includes a tool chain taking an MO2 architectural model and the referenced 20-sim specifications (defined in the SIDOPS+ language) as input. The tool chain contains a graphical editor, which is an extension of the ArchStudio 4 tool set [8], to create and edit MO2 architectural models. The *MO2 consistency validator* checks the consistency of the MO2 model. For example, it checks whether each variable that is used in constraints and objective functions has a mathematical relationship with the decision variables. Such a relationship should have been specified using 20-sim models. If the MO2 architectural model is consistent, it can be provided to the *MO2 code generator*, to generate an optimiser module specific for the given architecture and MO2 model. The software modules that implement the basic control architecture are provided to the *MO2 code weaver*. The code weaver introduces the interaction between these software modules and the generated optimiser module by weaving instrumentation code in the software modules. The result is embedded control software that includes multi-objective optimisation functionality.

The following section introduces an industrial case study, which will be used in subsequent sections to illustrate the three steps.

8.4.2 Industrial Case Study

In this section, we present an industrial case study where we have applied our MO2 method to a digital printing system, in particular to the subsystem related to the *Warm Process* of professional printers.[2] This process, schematically shown in Fig. 8.13, is responsible for transferring a *toner image* to paper: The *paper path*

[2]A video demonstrating the application of our method in the case study can be found at the Octopus homepage of the University of Twente: http://www.utwente.nl/ewi/trese/research_projects/Octopus.doc/.

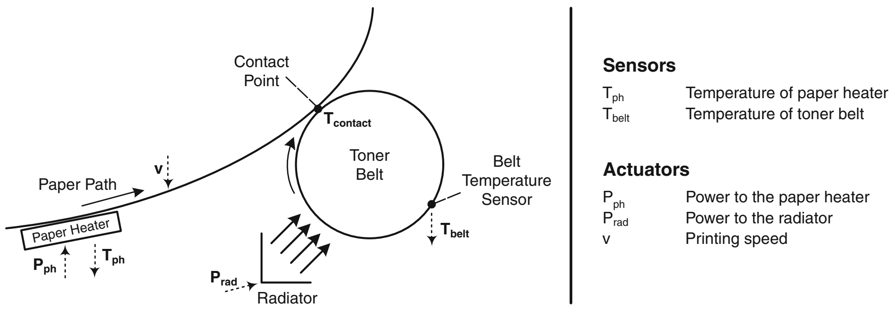


Fig. 8.13 Schematic view of the *Warm Process*

transports sheets of paper and a *toner belt* to transport toner images. For correct printing, the sheets of paper and the toner belt must have a certain temperature at the contact point. Therefore, the warm process contains two heating systems; a *paper heater* to heat the sheets of paper and a *radiator* to heat the toner belt.

8.4.2.1 Multi-Objective Optimisation in the Warm Process

In this case study, engineers aim to introduce the possibility to make run-time trade-offs between the two conflicting objectives *power consumption* and *productivity* of the printing system. They have identified the decision variables that can be used to influence the objectives, the different constraints in the system, and the objective functions:

- **Decision Variables:** Paper transportation speed (v).
- **Constraints:**
  1. $60 \leq v \leq 120$ (Speed between 60 and 120 pages/min).
  2. $P_{\text{rad}} \leq 800$ (Maximum power to the radiator is 800 W).
  3. $P_{\text{ph}} \leq 1{,}200$ (Maximum power to the paper heater is 1,200 W).
  4. $P_{\text{total}} \leq P_{\text{avail}} - 100$ (Total power consumption should not exceed the amount of power that is available to the system, and a margin of 100 W is taken into account).
  5. $40 \leq T_{\text{ph}}^{\text{sp}} \leq 90$ (Setpoint for the paper heater is between 40 and 90° C).
- **Objective functions:**
  - Minimisation of total power consumption: $P_{\text{total}} = P_{\text{ph}} + P_{\text{rad}}$.
  - Minimise the inverse of the speed (for an increased productivity): $1/v$.

  A trade-off between these objectives is made using a weighted trade-off function. Since the trade-off function can change, e.g. due to user requests, this problem is different from single-objective optimisation.

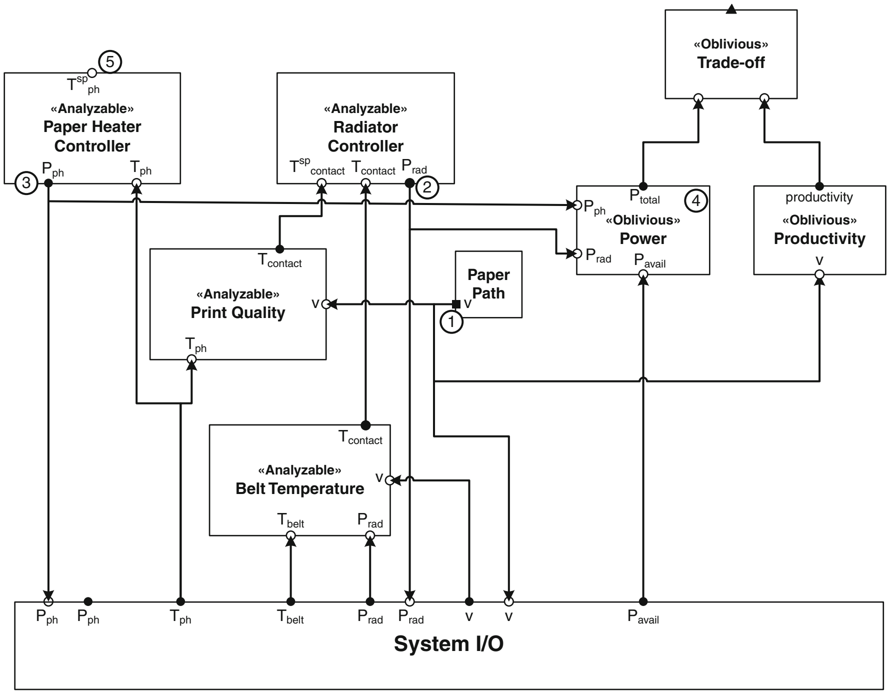


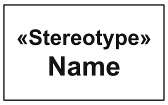
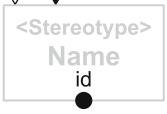
Fig. 8.14 MO2 model of the case study

8.4.2.2 MO2 Architectural Model

Control software implements the control logic for this system. To implement multi-objective optimisation, an MO2 architectural model of the control software has been created. Figure 8.14 shows the graphical representation of this MO2 model. The notation, i.e. our domain-specific modelling language for the MO2 style, is briefly explained in Table 8.1.

Basic control logic is implemented in several components in the control software. The `Paper Heater Controller` controls the temperature of the paper heater (T_{ph}) to a certain setpoint (T^{sp}_{ph}), by adjusting the power given to the paper heater (P_{ph}). The `Radiator Controller` controls the temperature of the toner belt at the contact point ($T_{contact}$) to a certain setpoint ($T^{sp}_{contact}$), by adjusting the power given to the radiator (P_{rad}). The value of $T^{sp}_{contact}$ is provided by the `Print Quality` component. The `Print Quality` component implements a model of print quality, which relates the value of $T_{contact}$ to the values of v and T_{ph}, to ensure sufficient print quality. Because there is no sensor in the system to measure the value of $T_{contact}$, the value of $T_{contact}$ is derived from the values of T_{belt}, v, and P_{rad} using a physical model of the belt temperature that is implemented in the `Belt Temperature` component.

Table 8.1 Model elements of the MO2 style notation

Notation	Description
	Component with a *stereotype* and a *name*. Here, the stereotype `Analyzable` indicates that there is a mathematical model of (part of) the semantics of the component. The stereotype `Oblivious` indicates that the component is only present to support the modelling of the multi-objective optimisation problem; it is not implemented.
○ ●	In-port/out-port
□ ■	In-port/out-port that represents a decision variable.
▽ ▼	In-port/out-port that represents an objective value.
	Usage of a port: The ports that belong to a component are attached to the edge of the component. The port may be labelled with its `variableName`.
→	Connector
①	Informal label indicating constraints attached to the component or port.

The `Paper Path` component provides the default speed (v). The `System I/O` component provides an interface to the sensors and actuators in the system.

The modelled components `Power`, `Productivity`, and `Trade-Off` have the stereotype `Oblivious`, which means that they are not implemented in the control software; these components are modelled to specify parts of the multi-objective optimisation problem. The specification for the example in Fig. 8.14 is as follows:

- The out-port v of the `Paper Path` component is a *decision variable port*, specifying that v is a decision variable.
- The labels 1–5 indicate constraint ports; the corresponding constraints are not shown in the figure.
- The two objective functions for power consumption and productivity are modelled using the `Power` and `Productivity` components. The trade-off function is modelled using the `Trade-Off` component. The out-port of this component is the only *objective port* in the system, indicating that the optimiser should minimise this value.

8.4.2.3 20-Sim Models of Control Logic

To solve an multi-objective optimisation problem, the mathematical relationship between the decision variables, and the constraints and objective functions must be known. The reason for this is that only when this mathematical relationship is known, it can be analysed which values should be selected for the decision variables to satisfy the constraints and optimise the objective functions. The mathematical relationship between the decision variables and the different constraints and objective functions in an MO2 model is defined by the semantics of the control components in the MO2 model. To analyse the specified MO2 model and to generate an optimiser, this semantics should be available to the MO2 tool chain. Therefore, the components in an MO2 model can reference a 20-sim [1, 3] model that specifies that part of the component's semantics that is relevant for creating the mathematical relationship.

Listings 8.6 and 8.7 show two example 20-sim models (using the SIDOPS+ language [22] of the 20-sim tool set) for respectively the components `Radiator Controller` and `Power`. Similar 20-sim models are referenced by the other components with stereotype `Analyzable` or `Oblivious` in the MO2 model.

Listing 8.6 SIDOPS+ specification referenced by the `RadiatorController` component.
```
constants
    real Kp_rad = #some value#;
    real Ki_rad = #some value#;
variables
    real global Tcontact;
    real global TcontactSP;
    real global Prad;
equations
    Prad = Kp_rad * (TcontactSP - Tcontact) + Ki_rad *
        int(TcontactSP - Tcontact);
```

Listing 8.7 SIDOPS+ specification referenced by the `Power` component.
```
variables
    real global Pph;
    real global Prad;
    real global Ptotal;
    real global Pavailable;
equations
    Ptotal = Pph + Prad;
```

8.4.2.4 Analysis and Code Generation

The MO2 tool chain can analyse whether for all constraints and objective functions there is a mathematical relationship with the decision variables. For example, for the MO2 model in Fig. 8.14 the tooling detects that there is no mathematical relationship between the decision variable v and the constraints on the ports P_{ph} and T_{ph}^{sp} (labelled 3 and 5) on the component `Paper Heater Controller`. This means that the optimiser cannot influence these constraints, and they are ignored.

The code generator in the MO2 tool chain generates an optimiser component and interaction with the implementation of other control modules. Figure 8.15 shows the structure of the implementation, including the generated optimiser. Note the absence of the `Oblivious` components `Power`, `Productivity`, and `Trade-off`: The only function of these components was to model the multi-objective optimisation problem. Therefore, they are not present in the implementation of the system; their semantics is part of the generated optimiser component.

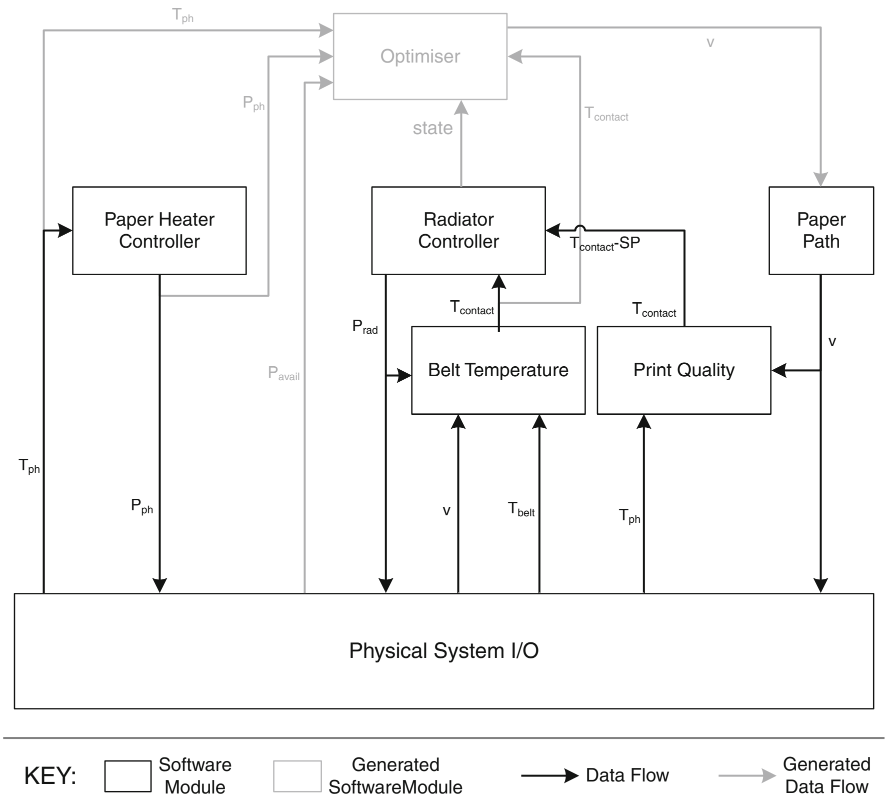


Fig. 8.15 Software structure with generated optimiser

8.4.3 Evaluation

In this section we compare the control implementation that contains multi-objective optimisation (MO2 implementation), as presented in the previous sections, with the control implementation that contains a state-of-the-practise algorithm to optimise productivity, called *Intelligent Speed*.

8.4.3.1 Intelligent Speed Algorithm

Algorithm 8.1 shows the Intelligent Speed algorithm. The Intelligent Speed algorithm works as follows. The amount of power that is not utilised (P_{margin}) is calculated. When P_{margin} it too low, the speed is decreased with 20 ppm. When P_{margin} is higher than a certain boundary (200 W), then the speed is increased with an amount that is proportional to the actual value of P_{margin}.

Algorithm 8.1: Intelligent speed algorithm

```
1  $P_{\text{total}} := P_{\text{ph}} + P_{\text{rad}}$
2  $P_{\text{margin}} := P_{\text{avail}} - P_{\text{total}}$
3  **if** $P_{margin} \leq 0$ **then**
4      $v_{\text{new}} := v - 20$
5  **end**
6  **else if** $P_{margin} \geq 200$ **then**
7      $v_{\text{new}} := v + 0.05 \cdot P_{\text{margin}}$
8  **end**
9  **else**
10     $v_{\text{new}} := v$
11 **end**
   // Ensure that new speed is within limits:
12 $v_{\text{new}} := \min(120, \max(60, v_{\text{new}}))$
```

Note that this algorithm does not optimise the power margin P_{margin} to 0 W, but maintains a certain amount of margin, which varies between 0 W and 200 W. This margin is used in real printer systems to cope with a delay in which the speed can be changed; in this experiment we assume that speed can be changed instantaneously, but in real printer systems there is a delay of a number of seconds before the speed can be adapted. To cope with sudden drops in the amount of power available during this delay period, the algorithm maintains a margin.

To make a fair comparison between the MO2 implementation and the Intelligent Speed implementation, the MO2 implementation maintains a power margin of 100 W. This is implemented by specifying the constraint $P_{\text{total}} \leq P_{\text{avail}} - 100$ on the P_{total} out-port of the `Power` component, as was demonstrated in Sect. 8.4.2.

8.4.3.2 Comparison

The performance of the two control software implementations regarding productivity is measured in an experimental setup. This setup uses a MATLAB/Simulink model of the Warm Process thermodynamics, to simulate the Warm Process part of the physical printer system. This Simulink [23] model has been provided by our industrial partner Océ, and is a realistic model of a printer system.

Several scenarios have been simulated. Each scenario has a fluctuating, but limited, amount of power available. Scenarios with different power fluctuation intervals (i.e. the interval after which the amount of available power changes) have been simulated. Each scenario runs for 20,000 time steps (simulated seconds).

Figure 8.16 provides the average printing speed obtained by the two implementations for each power fluctuation interval. This figure clearly shows that the MO2 implementation provides higher productivity than the Intelligent Speed algorithm.

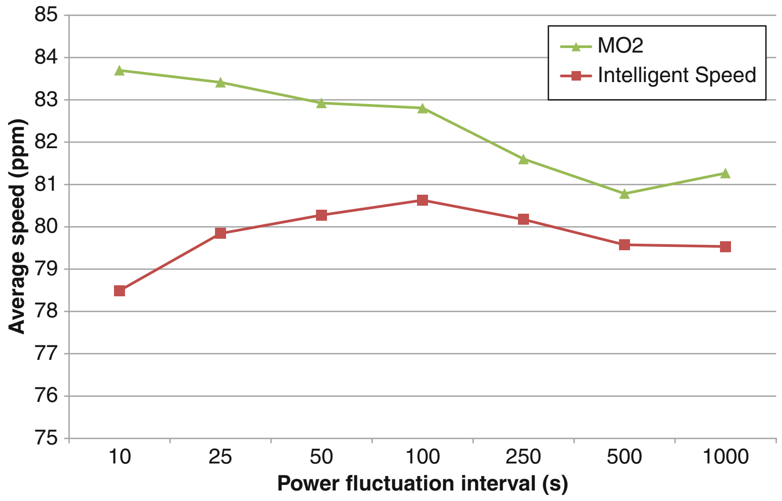


Fig. 8.16 Average printing speed

Figure 8.17 shows the average power margin obtained by the two implementations for each power fluctuation interval. As explained before, the Intelligent Speed algorithm maintains a power margin between 0 W and 200 W, while the MO2 implementation tries to maintain a power margin of 100 W.

The figure shows that the Intelligent Speed implementation generally results in an average power margin above 100 W. The MO2 implementation results in a power margin that is slightly below 100 W. Especially for larger power fluctuation intervals, the average power margin of the MO2 implementation approaches 100 W. The average power margin of the MO2 implementation is generally below 100 W, because there are situations in which there is insufficient power available to print at the lowest speed and still maintain a 100 W margin. In this case, the constraint to maintain a 100 W margin is dropped.[3] These situations happen more often with higher power fluctuation intervals, as Fig. 8.17 shows.

Figure 8.18 shows the power margin of the Intelligent Speed algorithm during one simulated scenario. The figure shows that the power margin is not constant, but varies between 0 W and 200 W. This is inherent in the design of the Intelligent Speed algorithm (Algorithm 8.1 on page 274): the speed is increased when there is more than 200 W power margin and decreased when there is less than 0 W power margin. A problem with this design is that sometimes there is a low power margin

[3]This is a property of the chosen optimisation algorithm: When the solution space of the MOO problem is empty, this algorithm drops constraints until there is a solution. Additional configuration of this algorithm is done so that it drops the 100 W power margin constraint first. This property is not part of the MO2 method; the method leaves open what should happen when the solution space of the MOO problem is empty.

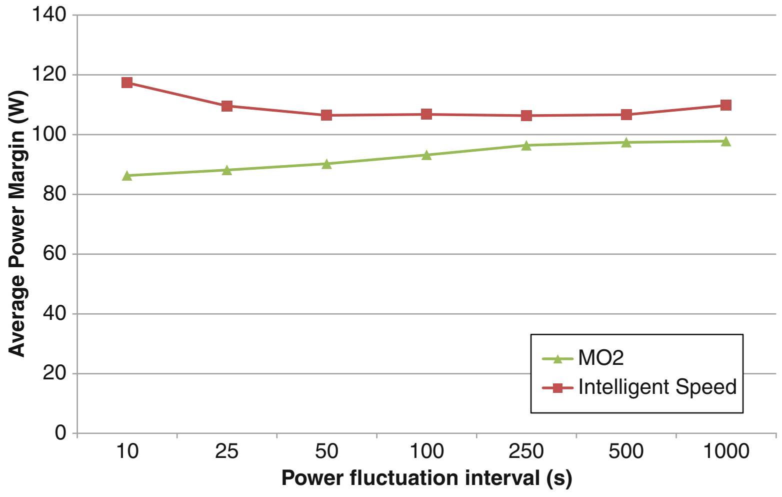


Fig. 8.17 Average power margin

available, making it harder to cope with sudden drops in the amount of power available. At other times there is a too high power margin available, which reduces productivity.

Figure 8.19 shows the power margin of the MO2 implementation for the same scenario as was used to demonstrate the power margin of the Intelligent Speed implementation in Fig. 8.18. Figure 8.19 demonstrates that the MO2 method is able to provide a precise and stable power margin of 100 W, as opposed to the Intelligent Speed algorithm which gives an unpredictable power margin between 0 W and 200 W. The short peaks and drops visible in Fig. 8.19 are caused by large changes in the amount of power available that the system cannot directly adapt to. The two longer drops in the power margin (around $0.65 \cdot 10^4$ s and $1.65 \cdot 10^4$ s) are caused by the fact that during this period there is not enough power available to print at the lowest speed and maintain a 100 W power margin. In this case, the power margin is used to continue printing at the lowest speed.

8.4.4 *Discussion: Computational Performance of Multi-Objective Optimisation*

Applying multi-objective optimisation algorithms results in a better performing system, but can also lead to a considerable computational performance overhead in software. This concern is particularly relevant for embedded systems, as these systems generally tend to have limited processing power available. In the end, this concern is an engineering trade-off: The engineer has to decide whether to

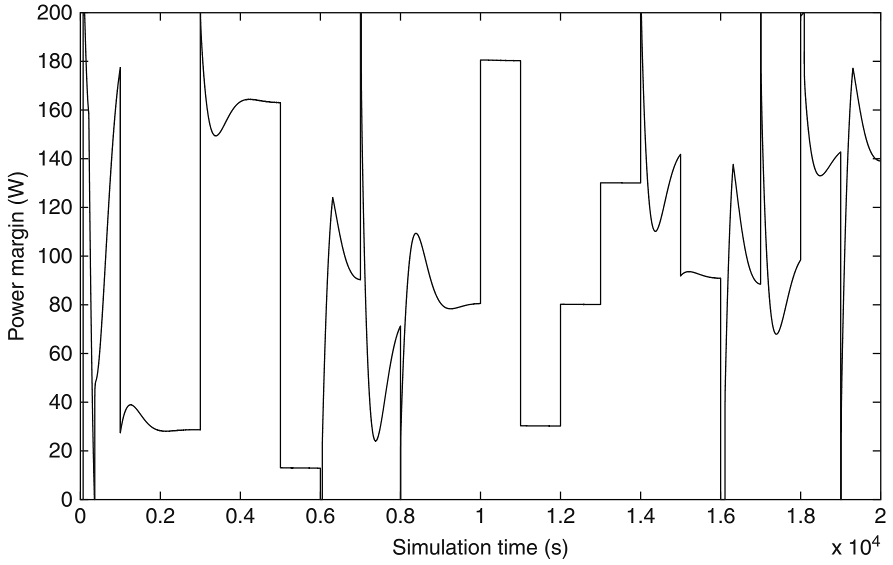


Fig. 8.18 Power margin during one scenario for the Intelligent Speed implementation

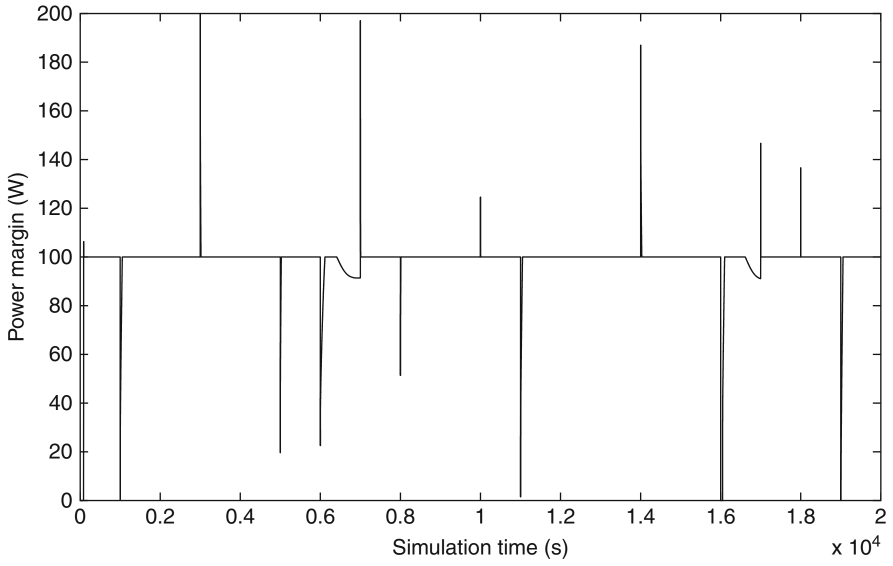


Fig. 8.19 Power margin during the same scenario for the MO2 implementation with 100 W margin

apply multi-objective optimisation algorithms for a better performing system, or not to apply them for more efficient software that can execute on limited processing hardware.

The computational complexity of solving a multi-objective optimisation problem depends on the characteristics of the problem. Certain multi-objective optimisation problems can be solved (deterministically) in polynomial time (e.g. linear programming problems [31]), while other problems are NP-hard [16]. As such, the trade-off can be influenced by careful selection of the multi-objective optimisation algorithm and adapting the multi-objective optimisation problem in such a way that it can be solved efficiently. For some systems an approximation of the optimal solution would be sufficient. In this case, using an approximation algorithm, instead of an algorithm that gives an exact result can reduce the processing power required by the optimisation software.

Many different multi-objective optimisation algorithms have been designed, both algorithms that give exact results and algorithms that give approximations. An extensive overview of multi-objective optimisation can be found in [13].

8.5 Conclusion

In this chapter we have argued that making the control component of embedded system software adaptive can lead to a competitive advantage. Without adaptive control, we typically have to statically constrain the control parameters conservatively to ensure that system constraints are always obeyed. In contrast, adapting control to the current context, e.g. environmental conditions, peer systems, or user requests, allows to reach the physical boundaries of the controlled system under specific dynamic conditions.

The research challenge here was to provide software engineers with a systematic approach to develop adaptive behaviour *without* compromising the quality of the software: While being able to manage the complexity, software must be able to adapt and evolve. To achieve this, software should be modular with limited dependencies between the modules. From a bird's eye view, our proposed approach consists of the following building blocks:

- Using domain-specific languages (DSLs) or domain-specific modelling languages (DSMLs) for expressing domain-specific behaviour efficiently. This reduces the complexity of components because developers can focus for one particular concern on *what* must be done by a component instead of *how* something is done.
- The declarative nature of DSLs and DSMLs also facilitates automatic processing of component definitions by tools; for example, such definitions can be analysed to check them for correctness, feasibility, or performance characteristics. Another example is automatic code generation; this is also suitable to establish

links between domain-specific components and the rest of the system without introducing strong dependencies between the source code of those components.

We have demonstrated our approach by means of two industrial case studies. The first example was performed in the domain of schedulers which compute an ordering in which to execute tasks to optimise the system performance according to a given objective. In a case study we have demonstrated our Scheduling Workbench for implementing schedulers in a domain-specific language separately from the system which is to be scheduled. The code generation of the workbench allowed later integration of the scheduler into a system. We have demonstrated that our approach reduces the development effort of evolving the scheduler definition and the development effort of replacing the scheduler definition completely.

In the second case study we have made the control in the Warm Process adaptive, i.e. the continuous control of a physical subsystem in a high-quality digital printer. We have presented the MO2 method and tool chain to dynamically optimise control according to multiple objectives, i.e. different quality characteristics. In the concrete example, we have applied this approach to optimise the printer system with respect to the trade-off between throughput and energy consumption. We have shown, that our approach indeed can lead to a better performing control system; in addition, our approach gives engineers the opportunity to replace (also at a later stage) the optimisation algorithm itself.

The key messages of our contribution are:

1. The software evolvability of a system is increased by separating the implementation of a (control) problem from other functionality.
2. The complexity of such an implementation can be kept low by using domain-specific (modelling) languages for expressing the (control) component.
3. The usage of DSLs and DSMLs enables static analysis as well as automatic generation of run-time support for the component and its integration with the rest of the system.
4. The integration of a component with the system can be achieved through so-called code weavers, a technology adopted from aspect-oriented programming. This technology links the execution of the component but shields the developer from having to deal with strong dependencies in the source code.
5. A methodology to get software specifications that are (relatively) cleanly modularised in the implementation, resulting in systems which are easier to understand and to maintain.

Acknowledgments This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. 20-sim tooling. http://www.20sim.com. Accessed Aug 2012
2. Banker, R.D., Datar, S.M., Kemerer, C.F., Zweig, D.: Software complexity and maintenance costs. Commun. ACM **36**, 81–94 (1993)
3. Broenink, J.F.: Modelling, simulation and analysis with 20-sim. Journal A **38**, 22–25 (1997)
4. Brucker, P.: Scheduling Algorithms, 3rd edn. Springer, Berlin (2001)
5. Clements, P., Bachman, F., Bass, L., Ivers, D.G.J., Little, R., Nord, R., Stafford, J.: Documenting Software Architectures: Views and Beyond. Addison-Wesley, Boston (2002)
6. Collette, Y., Siarry, P.: Multiobjective Optimization: Principles and Case Studies. Springer, Berlin (2003)
7. Czarnecki, K.: Overview of generative software development. In: J.P. Banâtre, P. Fradet, J.L. Giavitto, O. Michel (eds.) Unconventional Programming Paradigms. Lecture Notes in Computer Science, vol. 3566, pp. 326–341. Springer, Heidelberg (2005)
8. Dashofy, E., Asuncion, H., Hendrickson, S., Suryanarayana, G., Georgas, J., Taylor, R.: Archstudio 4: An architecture-based meta-modelling environment. In: Companion to the Proceedings of the 29th International Conference on Software Engineering (ICSE'07), Minneapolis, pp. 67–68 (2007)
9. de Roo, A., Sözer, H., Akşit, M.: An architectural style for optimizing system qualities in adaptive embedded systems using multi-objective optimization. In: Proceedings of the 8th Working IEEE/IFIP Conference on Software Architecture (WICSA 2009), Cambridge, pp. 349–352 (2009)
10. de Roo, A., Sözer, H., Akşit, M.: Runtime verification of domain-specific models of physical characteristics in control software. In: Proceedings of the Fifth IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI 2011), Dallas, pp. 41–50 (2011)
11. de Roo, A.J.: Managing software complexity of adaptive systems. Ph.D. thesis, University of Twente, Enschede (2012)
12. Edgeworth, F.Y.: Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences. C. Kegan Paul, London (1881)
13. Ehrgott, M., Gandibleux, X. (eds.): Multiple criteria optimization: state of the art annotated bibliographic surveys. International Series in Operations Research & Management Science, vol. 52. Kluwer Academic, Dordrecht (2002)
14. Elrad, T., Fillman, R.E., Bader, A.: Aspect-oriented programming. Commun. ACM **44**, 29–32 (2001)
15. Filman, R.E., Elrad, T., Clarke, S., Akşit, M. (eds.): Aspect-Oriented Software Development. Addison-Wesley, Boston (2005)
16. Glaßer, C., Reitwießner, C., Schmitz, H., Witek, M.: Approximability and hardness in multi-objective optimization. In: Programs, Proofs, Processes. Lecture Notes in Computer Science, vol. 6158, pp. 180–189. Springer, Heidelberg (2010)
17. Hatley, D.J., Pirbhai, I.A.: Strategies for real-time system specification. Dorset House, New York (1987)
18. Hatun, K., Bockisch, C., Sözer, H., Akşit, M.: A feature model and development approach for schedulers. In: Proceedings of the 1st Workshop on Modularity in Systems Software (MISS 2011), Porto de Galinhas, pp. 1–5 (2011)
19. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. Tech. Rep. CMU/SEI-90-TR-21, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA (1990)
20. Keeney, R.L., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Wiley, New York (1976)
21. Kent, S.: Model driven engineering. In: M. Butler, L. Petre, K. Sere (eds.) Integrated Formal Methods. Lecture Notes in Computer Science, vol. 2335, pp. 286–298. Springer, Berlin (2002)
22. Kleijn, C.: 20-sim 4.1 Reference Manual (2009)

23. MATLAB/Simulink (2010). http://www.mathworks.com/products/simulink/. Accessed May 2012
24. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37**, 316–344 (2005)
25. Object Management Group: OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2 (2007). http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF. Accessed Aug 2012
26. Pareto, V.: Cours D'Économie Politique. F. Rouge, Lausanne (1896)
27. Selic, B.: Using UML for modeling complex real-time systems. In: F. Mueller, A. Bestavros (eds.) Languages, Compilers, and Tools for Embedded Systems. Lecture Notes in Computer Science, vol. 1474, pp. 250–260. Springer, Berlin (1998)
28. van de Laar, P., Punter, T. (eds.): Views on Evolvability of Embedded Systems. Springer, Dordrecht (2011)
29. van Engelen, R., Voeten, J. (eds.): Ideals: Evolvability of Software-Intensive High-Tech Systems. Embedded Systems Institute, Eindhoven (2007)
30. Ward, P.T., Mellor, S.J.: Structured development for real-time systems: Introduction & tools. Yourdon Press, Englewood Cliffs (1985)
31. Winston, W.L.: Operations research: applications and algorithms, 4th edn. Thomson Brooks/-Cole, Stamford (2004)

Chapter 9
Reflections on the Octopus Project

Frans Reckers, Twan Basten, Roelof Hamberg, and Jacques Verriet

Abstract How to develop adaptive high-tech embedded systems? This question has motivated this book. Adaptivity provides a wider operational envelope of systems and enables new market opportunities. Effective development of such systems requires a systematic approach that enables to include the adaptive capabilities at various system levels. This book presents the most important results of the Octopus project. Being an industry-as-laboratory project, Octopus brought together academic partners and engineers from Océ-Technologies. Industrial challenges in professional printing motivated the research and results were validated in industrial practice. This approach turned out to be successful, with results ranging from improved control schemes for adaptive inkjet printing to design-space exploration tooling for embedded electronics and architecting patterns for adaptive embedded systems. Octopus confirmed some important points of attention for industry-as-laboratory projects. First, the paradigm needs academic researchers with a strong motivation to improve the industrial state-of-the-art. Second, it is not always easy to combine progress of industrial practice with scientific progress. A crisp definition of industrially relevant and academically challenging driver cases is crucial.

F. Reckers (✉) • R. Hamberg • J. Verriet
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: frans.reckers@esi.nl; roelof.hamberg@esi.nl; jacques.verriet@esi.nl

T. Basten
Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Electronic Systems group, Faculty of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
e-mail: a.a.basten@tue.nl

9.1 Introduction

The Octopus project was a follow-up of the successful BodeRC project [1], which brought a number of results for Océ and promising new research directions. The “HappyFlow” model concerning the paper flow in printers has enabled to reduce the printer development time significantly, because the model allows virtual exploration of paper path designs.

The impact of this result alone saved many man-years of effort and reduced time-to-market of newly developed printers. The process of creating models however was not perceived as transparent. So, further research into transparent methods to construct models on different system levels constituted a promising new research direction. The use of models offers high potential savings in development effort and costs, and a further reduction in product creation lead-time.

The Octopus project was initiated to further explore the insights of BodeRC. The project’s research was focused on methods to design key properties of future printing systems that are expected to operate in a diverse and highly dynamic business environments, with continuously changing requirements for context, function, application, and technology. In order to ensure the competitive positioning of such systems in a global market, it has become of paramount importance to include the correct level of system adaptability in the design.

The objective of the Octopus project has been to find concepts, methods, and tooling for the design of printing systems with an adequate adaptability at system, subsystem, and component level. Central to these concepts is the transparent coordination of system control over different technology domains making use of multi-disciplinary models. This coordination should be supported by systems architectures that intrinsically facilitate adaptability. The resulting methods serve to incorporate run-time adaptability features in such a way that high-quality prints can be produced for a wide variety of media types (even new ones) under varying environmental conditions. Although the goals were made specific for the professional printing domain, they are relevant for the high-tech embedded systems industry in general.

Looking to the research content the subjects can be grouped in two categories of subjects, the first being control of embedded systems and the second being architecting and design of adaptive embedded systems. This division of subjects has been followed in the preceding chapters and is followed in this reflection chapter as well.

From an organisational point of view, the Octopus project was defined under the Dutch BSIK regulations as a research project in an industry-as-laboratory setting [2] with consortium partners Océ-Technologies B.V., the Embedded Systems Institute, Delft University of Technology, Eindhoven University of Technology, Radboud University Nijmegen, and the University of Twente.

This chapter reflects with a bird’s-eye view on the outcome of the Octopus project. Section 9.2 provides an overview of the most important results. A reflection on the impact and the key lessons learned is to be found in Sects. 9.3 and 9.4, respectively. Section 9.5 presents some conclusions.

9.2 Overview of Technical Results

We provide an overview of the results of the project, following the partitioning made in Chap. 1 into "control of adaptive embedded systems" and "architecting and design of adaptive embedded systems". Along this division, we discuss the industrial impact of the results.

9.2.1 Control of Adaptive Embedded Systems

Control of adaptive embedded systems was the topic of Chaps. 3–5. This section describes the main results of these chapters.

Adaptive control of inkjet printheads. In Chap. 3, different feedforward control schemes were presented. New model-based and experiment-based control schemes for jetting pulses have been developed for the single-channel and multiple-channel printhead use cases with considerable print quality improvements, beyond the current state-of-the-art. The main ingredient for these results is the obtained independence of the ink jetting process from its frequency of operation and the requested payload.

Adaptive control at system level. Chapter 4 presented and compared approaches to system-level adaptive control; this involved two approaches to model reference adaptive control (MRAC) and two for model predictive control (MPC). Each of these approaches has its own merits. Chapter 4 showed that the MRAC approaches offer a robust performance at low costs, whereas MPC allows the best performance at the cost of a higher computational load. Both approaches provide an improvement over the current state-of-the-art, as overall better trade-offs in the quality/time/cost triangle can be obtained.

Probabilistic graphical models for adaptive control. The fuse temperature is a key factor for the print quality and is determined by many mutually dependent parameters. A new control approach for the fuse temperature using Bayesian network models has been developed. It was presented in Chap. 5. This novel approach resulted in a patent application [3]. Another new Bayesian network model application involves a classifier that can discriminate between different print media through deduction from the behaviour of the print process.

9.2.2 Architecting and Design of Adaptive Embedded Systems

Chapters 6–8 discussed architecting and design of adaptive embedded systems. The main results of these chapters are listed in this section.

Systematic support for systems architecting. Chapter 6 presented a new approach and tooling to support the first phases of systems architecting. System specifications with physical phenomena lead to parameter network models that involve the key parameters of a system and their relations. This is the starting point for different embodiments of the system in terms of subsystems and architecture. Selected options can be analysed both qualitatively and quantitatively. The approach has been demonstrated with a first example, implemented with extensions of the KIEF prototype tooling [4].

Methodology and tool set for design-space exploration. Chapter 7 presented a tool set for design-space exploration. This Octopus tool set has been verified on several printer data path use cases. Different methods can be combined to generate visualisations of system performance metrics based on (graphical) system specifications. Because of the complex processing steps, industrially sound tooling is required. An open architecture has been designed for a tool set that allows flexible deployment, future changes, and incorporation of add-ons. The impact of combined application of the methodology and the tool set is high: improved accuracy in specifications, better understanding of system behaviour through visualisations, much faster exploration of design options, and improved documentation. All together this leads to faster system development.

Architecting and design of adaptive software. Chapter 8 considered model-based development of adaptive software applied to two industrial cases. One case involves a new method to construct the software architecture of a physical system with clear separation of software components dealing with system resources, system behaviour, and the control of system behaviour. An architectural style, called MO2 (Multi-Objective Optimisation), has been developed to structure the software of a system in which objective trade-offs must be made at runtime. A tool framework was developed and it was demonstrated how it supports software architecting. The other case concerns model-based design of scheduling systems. The chapter presented a scheduling specification language called scheDL and a supporting tool framework to accomplish executable scheduling models for simulation. The simulation results provide feedback on the performance of the proposed solutions to the engineer for possible improvements of the system scheduling policy.

9.3 Impact

This section summarises the impact of the Octopus project.

Strategic benefit. The project has provided Océ with an important early inflow of scientific knowledge that is required to tackle high-risk long-term industrial challenges with potential high impact. The project has brought new research problems to ESI and the academic partners. The leverage for all stakeholders justifies the total investments from an economic, societal, and financial perspective.

Improved design effectiveness. The emphasis on model-based design methods has enriched the existing basis of effective model-based design at Océ. New concepts and methods have been introduced to develop models for industrial cases such as data path design-space exploration, multi-channel inkjet control, and system-level printer control. Tooling has been built to support the introduction of the scientific concepts and methods for industrial application. The combination of methods and tools offers support to achieve high design quality, design speed, and improved team communication while design effort and thus lead-time and costs decrease.

Industrial embedding for education. The project has provided an excellent embedding for education of highly skilled professionals. The academic researchers and industrial researchers have cooperated on design-space exploration methods applied to printer data paths, control approaches applied to inkjet printheads, and methods for systems and software architecting applied to adaptive printer system control. A substantial body of knowledge has been developed by the academic project partners. There are 16 senior academic researchers, 8 postdoctorate researchers, and 7 PhD students of 7 distinct academic groups who have extended their scientific knowledge. One postdoctorate researcher has been awarded an academic "Veni" grant. A group of 16 PDeng students was guided on basis of an Octopus assignment (in total 2 man years of education effort), and 5 PDeng students conducted their final projects (each 9 months) in Octopus. The project has hosted 13 master's students for their master's theses with subjects instigated by the Octopus project research.

Scientific publications. Throughout the project the research team published more than 100 scientific publications including journal papers, conference contributions, technical reports, PhD theses, and master's theses. An overview of these publications can be found in Appendix B.

9.4 Reflection

The cooperation between Océ, ESI, and academic partners in industry-as-laboratory research projects like Octopus is evaluated in this section. Especially the learning points are discussed.

Focus on one integrated business case. It is needed to have a realistic and challenging use case as a carrier for intensive information exchange between the involved academic researchers and industrial developers. A proper problem should fit genuine business goals of the carrying industrial partner. Starting the project with finding one business case incorporating clear research questions with a proper scope that enforces cooperation across all members of the research team is therefore recommended. Spending sufficient time to find shared goals and to gain deeper understanding of each other's interests is needed. This stimulates cooperation between project partners across research lines to solve the "big" shared problem in an optimal way.

Requirements on personal skills. The recruitment of research staff for Octopus was an intensive process with much effort of the academic supervisors to find suitable candidates. We have learned that successful industry-as-laboratory projects need researchers who want to and are able to cross the borders of their own academic discipline to other academic and industrial disciplines: PhD students and postdoctorate researchers have to satisfy both academic and industrial goals with a suitable mix of qualities. This is the most prominent lesson learned for all involved project partners. It will lead us to the introduction of selection methods that go beyond the standard job interview.

Timely acquisition of domain knowledge. At the start of the project the research team had very little detailed knowledge of the domain in which they would be working. Investing time and effort of all partners to give the team a firm basis in the core technology and business goals of the carrying industrial partner is very important to put the research objectives into perspective. An internship of a few months at the start of the project is expected to substantially accelerate the acquisition of domain knowledge by the academic researchers. This hold primarily for the "embedded" researchers themselves, but to some degree also for their supervisors.

Management of disruptive technology changes. The Octopus project started with research topics that focused on "direct image printing" technology as the key business driver. During the project "inkjet technology" became the primary business driver. This kind of major technology change must be considered a "fact-of-life". The key learning point here is that we have to develop a better awareness that such disruptive changes may happen during the life-time of a project, with big impact for the research focus and activities. Such changes require activities that normally belong to the start-up period of a project, e.g. acquiring new domain knowledge. The adoption of new technologies in industry needs management focus on strong involvement of industrial (Océ) staff and strong interaction with academic researchers for real implementation.

Positioning of research subjects. Academic groups provide excellent and highly valuable new results to a project, as was the case for Octopus. Combining these new technologies into system-wide solutions, and integrating them into an architectural framework requires competences which not all academic researchers have. In order to make such complicated multi-disciplinary projects into a success, "systems architecting" must be incorporated as one of the key project components. Architecting considerations must be introduced ahead of – or in the earliest stages of – the project, as these must define the context, objectives, and boundaries of the individual academic partners. This allows integration of the contributions of each individual partner into a strong, integrated use case with high business relevance.

9.5 Concluding Remarks

This chapter concludes the book “Model-Based Design of Adaptive Embedded Systems”. The book summarises the achievements of the Octopus project. The project team took up the challenge to find answers to various aspects of the research question “How to develop adaptive embedded systems?”.

The value of the “industry-as-laboratory” project approach has been confirmed by the project outcomes for all partners involved in this research journey. The project leverage justifies the total investments from economic, societal, and financial perspective.

The concluding remark in the project’s final review meeting by the vice president of Océ responsible for the project nicely summarises the project and its impact: “We can be pleased with how far we got, but we want more! So we will extend activities on the topics touched upon in Octopus”. This is being done in Octopus’ successor project, Octo+,[1] the third industry-as-laboratory project of Océ-Technologies and the Embedded Systems Institute.

Acknowledgements This work has been carried out as part of the Octopus project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs, Agriculture, and Innovation under the BSIK program.

References

1. Heemels, M., Muller, G. (eds.): Boderc: Model-Based Design of High-Tech Systems. Embedded Systems Institute, Eindhoven (2006)
2. Potts, C.: Software-engineering research revisited. IEEE Softw. **10**, 19–28 (1993)
3. Waarsing, B.J.W., van den Bosch, P.F.A., Hommersom, A.J.: Reprographic system. Patent Application PCT/EP2009/066348 (2010)
4. Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., Tomiyama, T.: Physical concept ontology for the knowledge intensive engineering framework. Adv. Eng. Inform. **18**, 95–113 (2004)

[1]More information on the Octo+ project can be found on the project’s website: http://www.esi.nl/octoplus/.

Appendix A
Octopus Project Partners

This appendix provides an overview of the organisations that have participated in the Octopus project.

- Embedded Systems Institute, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Océ-Technologies B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands.
- Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, Delft Center for Systems and Control, P.O. Box 5, 2600 AA Delft, The Netherlands.
- Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, Department of BioMechanical Engineering, Intelligent Mechanical Systems group, P.O. Box 5, 2600 AA Delft, The Netherlands.
- Eindhoven University of Technology, Faculty of Electrical Engineering, Control Systems group, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Eindhoven University of Technology, Faculty of Electrical Engineering, Electronic Systems group, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Eindhoven University of Technology, Faculty of Mathematics and Computer Science, Architecture of Information Systems group, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
- Radboud University Nijmegen, Institute for Computing and Information Sciences, Department of Model-Based System Development, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands.
- University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, Software Engineering group, P.O. Box 217, 7500 AE Enschede, The Netherlands.

Appendix B
Octopus Project Publications

This appendix provides an overview of the articles, papers, reports, theses, and other publications published within the scope of the Octopus project.

[1] AlAttili, I., Houben, F., Igna, G., Michels, S., Zhu, F., Vaandrager, F.: Adaptive scheduling of data paths using Uppaal Tiga. In: Proceedings of the First Workshop on Quantitative Formal Methods: Theory and Applications (QFM2009), pp. 1–11 (2009)

[2] Arts, L.: Towards adaptable product architecture: A method to incorporate adaptability in the product architecture of complex products. Master's thesis, Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, Department of BioMechanical Engineering, Delft (2008)

[3] Arts, L., Chmarra, M.K., Tomiyama, T.: Modularization method for adaptable products. In: Proceedings of the ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2008), Paper no. DETC2008-49338, pp. 193–202 (2008)

[4] Basten, T., Hendriks, M., Somers, L., Trčka, N.: Model-driven design-space exploration for software-intensive embedded systems (extended abstract). In: M. Jurdzinski, D. Nickovic (eds.) Formal Modeling and Analysis of Timed Systems, *Lecture Notes in Computer Science*, vol. 7595, pp. 1–6. Springer, Berlin (2012)

[5] Basten, T., van Benthum, E., Geilen, M., Hendriks, M., Houben, F., Igna, G., Reckers, F., de Smet, S., Somers, L., Teeselink, E., Trčka, N., Vaandrager, F., Verriet, J., Voorhoeve, M., Yang, Y.: Model-driven design-space exploration for embedded systems: The Octopus toolset. In: T. Margaria, B. Steffen (eds.) Leveraging Applications of Formal Methods, Verification, and Validation, *Lecture Notes in Computer Science*, vol. 6415, pp. 90–105. Springer, Heidelberg (2010)

[6] Blokpoel, M., Kwisthout, J., van der Weide, T.P., van Rooij, I.: How action understanding can be rational, Bayesian and tractable. In: Proceedings of the 32nd Annual Meeting of the Cognitive Science Society (COGSCI 2010), pp. 1643–1648 (2010)

[7] Blokpoel, M., Kwisthout, J., Wareham, T., Haselager, P., Toni, I., van Rooij, I.: The computational costs of recipient design and intention recognition in communication. In: Proceedings of the 33rd Annual Meeting of the Cognitive Science Society (COGSCI 2011), pp. 465–470 (2011)

[8] Brandt, R.: Reduction of effects of residual dynamics and cross coupling in drop-on-demand inkjet printhead. Master's thesis, Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, Delft Center for Systems and Control, Delft (2010)

[9] Chen, X.: DSET: Design-Space Exploration Toolset. Tech. rep., Stan Ackermans Institute / Software Technology, Eindhoven (2009)

[10] Chmarra, M.K., Álvarez Cabrera, A.A., van Beek, T., D'Amelio, V., Erden, M.S., Tomiyama, T.: Revisiting the divide and conquer strategy to deal with complexity in product design. In: Proceedings of the IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA2008), pp. 393–398 (2008)

[11] Chmarra, M.K., Arts, L., Tomiyama, T.: Towards adaptable architecture. In: Proceedings of the ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2008), Paper no. DETC2008-49971, pp. 367–376 (2008)

[12] Chmarra, M.K., Arts, L., Tomiyama, T.: Towards design-time and runtime adaptability. In: Proceedings of the 6th International Seminar and Workshop on Engineering Design in Integrated Product Development (EDIProD 2008), pp. 33–40 (2008)

[13] Chmarra, M.K., Verriet, J., Waarsing, R., Tomiyama, T.: State transition in reconfigurable systems. In: Proceedings of the ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2010), Paper no. DETC2010-28723, pp. 241–248 (2010)

[14] Cochior, C., van den Bosch, P., Waarsing, R., Verriet, J.: Optimal control of bilinear systems with time-varying constraints. In: Proceedings of the 30th Benelux Meeting on Systems and Control, p. 200 (2011)

[15] Cochior, C., van den Bosch, P., Waarsing, R., Verriet, J.: Control strategy for print quality control. In: Proceedings of the 31st Benelux Meeting on Systems and Control, p. 142 (2012)

[16] Cochior, C., van den Bosch, P.P.J., Waarsing, R., Verriet, J.: Cold start control of industrial printers. In: Proceedings of the 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2012), pp. 123–128 (2012)

[17] Cochior, C., van den Bosch, P.P.J., Waarsing, R., Verriet, J.: Control strategy for systems with input-induced nonlinearities: A printing system case study. In: Proceedings of the 2012 American Control Conference (ACC 2012), pp. 1949–1954 (2012)

[18] D'Amelio, V., Chmarra, M.K., Tomiyama, T.: Early design interference detection based on qualitative physics. Res. Eng. Des. **22**, 223–243 (2011)

[19] D'Amelio, V., Chmarra, M.K., Tomiyama, T.: A method to reduce ambiguities of qualitative reasoning for conceptual design applications. Artif. Intel. Eng. Des. Anal. Manuf. (2012). Accepted for publication

[20] de Roo, A., Hendriks, M., Havinga, W., Dürr, P., Bergmans, L.: Compose*: a language- and platform-independent aspect compiler for composition filters. In: Proceedings of the First International Workshop on Advanced Software Development Tools and Techniques (WASDETT2008), Paper no. 10 (2008)

[21] de Roo, A., Sözer, H., Akşit, M.: An architectural style for optimizing system qualities in adaptive embedded systems using multi-objective optimization. In: Proceedings of the 8th Working IEEE/IFIP Conference on Software Architecture (WICSA2009), pp. 349–352 (2009)

[22] de Roo, A., Sözer, H., Akşit, M.: Runtime verification of domain-specific models of physical characteristics in control software. In: Proceedings of the Fifth IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI 2011), pp. 41–50 (2011)

[23] de Roo, A., Sözer, H., Akşit, M.: Verification and analysis of domain-specific models of physical characteristics in embedded control software. Inform. Softw. Tech. **54**, 1432–1453 (2012)

[24] de Roo, A.J.: Managing software complexity of adaptive systems. Ph.D. thesis, University of Twente, Enschede (2012)

[25] de Vries, M.: Control of a drop-on-demand printhead. Master's thesis, Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, Delft Center for Systems and Control, Delft (2011)

[26] Evers, S., Lucas, P.J.F.: Variable elimination by factor indexing. In: Proceedings of the Fifth European Workshop on Probabilistic Graphical Models (PGM-2010), pp. 129–137 (2010)

[27] Evers, S., Lucas, P.J.F.: Constructing Bayesian networks for linear dynamic systems. In: Proceedings of the 8th UAI Bayesian Modeling Applications Workshop (BMAW-11), pp. 26–33 (2011)
[28] Evers, S., Lucas, P.J.F.: Marginalization without summation exploiting determinism in factor algebra. In: W. Liu (ed.) Symbolic and Quantitative Approaches to Reasoning with Uncertainty, *Lecture Notes in Computer Science*, vol. 6717, pp. 251–262. Springer, Heidelberg (2011)
[29] Ezzeldin, M., Jokic, A., van den Bosch, P.: Modeling and control of inkjet printhead. In: Proceedings of the 28th Benelux Meeting on Systems and Control, p. 68 (2009)
[30] Ezzeldin, M., Jokic, A., van den Bosch, P.: Model-free feedforward control of inkjet printhead. In: Proceedings of the 29th Benelux Meeting on Systems and Control, p. 36 (2010)
[31] Ezzeldin, M., Jokic, A., van den Bosch, P.P.J.: A parameter varying Lyapunov function approach for tracking control for Takagi Sugeno class of nonlinear systems. In: Proceedings of the 8th IEEE International Conference on Control & Automation (ICCA '10), pp. 428–433 (2010)
[32] Ezzeldin, M., Jokic, A., van den Bosch, P.P.J., Waarsing, R.: Model-free optimization based feedforward control for an inkjet printhead. In: Proceedings of the 2010 IEEE International Conference on Control Applications (CCA 2010), pp. 967–972 (2010)
[33] Ezzeldin, M., van den Bosch, P.P.J., Waarsing, R.: Improved convergence of MRAC design for printing system. In: Proceedings of the 2009 American Control Conference (ACC 2009), pp. 3232–3237 (2009)
[34] Ezzeldin, M., van den Bosch, P.P.J., Weiland, S.: Improving the performance of an inkjet printhead using model predictive control. In: Proceedings of the 18th IFAC World Congress (IFAC 2011), pp. 11544–11549 (2011)
[35] Ezzeldin, M., van den Bosch, P.P.J., Weiland, S.: Inverse-based feedforward control for an inkjet printhead. In: Proceedings of the 2011 American Control Conference (ACC 2011), pp. 4087–4092 (2011)
[36] Ezzeldin, M., van den Bosch, P.P.J., Weiland, S.: Towards a better printing quality for an inkjet printhead. IEEE Control Syst. **33**(1), pp. 42–60 (2013)
[37] Ezzeldin, M., Weiland, S., van den Bosch, P.: Gain scheduled static output feedback tracking control for a class of nonlinear systems via Takagi-Sugeno model. In: Proceedings of the 30th Benelux Meeting on Systems and Control, p. 173 (2011)
[38] Ezzeldin, M., Weiland, S., van den Bosch, P.P.J.: Improving the printing quality of an inkjet printhead using MIMO model predictive control. In: Proceedings of the 2011 IEEE International Conference on Control Applications (CCA 2011), pp. 382–387 (2011)
[39] Ezzeldin, M., Weiland, S., van den Bosch, P.P.J.: Robust L2 control for a class of nonlinear systems: a parameter varying Lyapunov function approach. In: Proceedings of the 19th Mediterranean Conference on Control and Automation (MED'11), pp. 213–218 (2011)
[40] Ezzeldin, M., Weiland, S., van den Bosch, P.P.J.: Improving the performance of a printing system using model reference adaptive control: An LMI approach. In: Proceedings of the 2012 American Control Conference (ACC 2012), pp. 1943–1948 (2012)
[41] Ezzeldin, M., Weiland, S., van den Bosch, P.P.J.: Observer-based robust L2 control for a professional printing system. In: Proceedings of the 2012 American Control Conference (ACC 2012), pp. 1955–1960 (2012)
[42] Ezzeldin Mahdy Abdelmonem, M.: Performance improvement of professional printing systems: from theory to practice. Ph.D. thesis, Eindhoven University of Technology, Eindhoven (2012)
[43] Ferreira, N., Hommersom, A., Lucas, P.: From probabilistic horn logic to chain logic. In: Proceedings of the 20th Belgian-Netherlands Conference on Artificial Intelligence (BNAIC 2008), pp. 73–80 (2008)
[44] Groot, P., Birlutiu, A., Heskes, T.: Learning from multiple annotators with Gaussian processes. In: T. Honkela, W. Duch, M. Girolami, S. Kaski (eds.) Artificial Neural Networks

and Machine Learning - ICANN 2011, *Lecture Notes in Computer Science*, vol. 6792, pp. 159–164. Springer, Heidelberg (2011)

[45] Groot, P., Lucas, P., van den Bosch, P.: Multiple-step time series forecasting with sparse Gaussian processes. In: Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC 2011), pp. 105–112 (2011)

[46] Hatun, K., Bockisch, C., Sözer, H., Akşit, M.: A feature model and development approach for schedulers. In: Proceedings of the 1st Workshop on Modularity in Systems Software (MISS 2011), pp. 1–5 (2011)

[47] Hendriks, M., Geilen, M., Basten, T.: Pareto analysis with uncertainty. In: Proceedings of the 9th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2011), pp. 189–196 (2011)

[48] Hendriks, M., Geilen, M., Basten, T.: Pareto analysis with uncertainty. ES Report ESR-2011-01, Eindhoven University of Technology, Department of Electrical Engineering, Electronic Systems group, Eindhoven (2011)

[49] Hendriks, M., Vaandrager, F.W.: Reconstructing critical paths from execution traces. In: Proceedings of the 10th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2012) (2012)

[50] Hommersom, A., Ferreira, N., Lucas, P.J.F.: Integrating logical reasoning and probabilistic chain graphs. In: W. Buntine, M. Grobelnik, D. Mladenić, J. Shawe-Taylor (eds.) Machine Learning and Knowledge Discovery in Databases, *Lecture Notes in Computer Science*, vol. 5781, pp. 548–563. Springer, Heidelberg (2009)

[51] Hommersom, A., Lucas, P.J.F.: Using Bayesian networks in an industrial setting: Making printing systems adaptive. In: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10), pp. 401–406 (2010)

[52] Hommersom, A., Lucas, P.J.F., Waarsing, R., Koopman, P.: Applying Bayesian networks for intelligent adaptable printing systems. In: Proceedings of the 7th Workshop on Intelligent Solutions in Embedded Systems (WISES 2009), pp. 127–133 (2009)

[53] Hommersom, A., Lucas, P.J.F., Waarsing, R., Koopman, P.: Applying Bayesian networks for intelligent adaptable printing systems. In: Proceedings of the 21st Benelux Conference on Artificial Intelligence (BNAIC 2009), pp. 443–444 (2009)

[54] Hommersom, A., Lucas, P.J.F., Waarsing, R., Koopman, P.: Applying Bayesian networks for intelligent adaptable printing systems. In: M. Conti, S. Orcioni, N. Martínez Madrid, R.E.D. Seepold (eds.) Solutions on Embedded Systems, *Lecture Notes in Electrical Engineering*, vol. 81, pp. 201–213. Springer, Heidelberg (2011)

[55] Houben, F., Igna, G., Vaandrager, F.: Modeling task systems using parameterized partial orders. In: Proceedings of the 18th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2012), pp. 317–327 (2012)

[56] Houben, F., Igna, G., Vaandrager, F.: Modeling task systems using parameterized partial orders. Int. J. Softw. Tools Technol. Transf. (2012). Accepted for publication

[57] Houben, F.H.M.: Design space exploration with generated timed automata. Master's thesis, Radboud University Nijmegen, Institute for Computing and Information Sciences, Nijmegen (2010)

[58] Igna, G., Kannan, V., Yang, Y., Basten, T., Geilen, M., Vaandrager, F., Voorhoeve, M., de Smet, S., Somers, L.: Formal modeling and scheduling of data paths of digital document printers. In: F. Cassez, C. Jard (eds.) Formal Modeling and Analysis of Timed Systems, *Lecture Notes in Computer Science*, vol. 5215, pp. 170–187. Springer, Heidelberg (2008)

[59] Igna, G., Vaandrager, F.: Verification of printer datapaths using timed automata. In: T. Margaria, B. Steffen (eds.) Leveraging Applications of Formal Methods, Verification, and Validation, *Lecture Notes in Computer Science*, vol. 6416, pp. 412–423. Springer, Heidelberg (2010)

[60] in 't Groen, A.M.: VDSEIR - A graphical layer on top of the Octopus toolset. Master's thesis, Eindhoven University of Technology, Faculty of Mathematics and Computer Science, Eindhoven (2011)

[61] Kannan, V., Somers, L., Voorhoeve, M.: Datapath architecture simulation. In: Proceedings of the 23rd European Simulation and Modelling Conference (ESM'2009), pp. 238–242 (2009)

[62] Kannan, V., van der Aalst, W.M.P., Voorhoeve, M.: Formal modeling and analysis by simulation of data paths in digital document printers. In: Proceedings of the Ninth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2008), pp. 27–45 (2008)

[63] Khalate, A.A., Bayon, B., Bombois, X., Scorletti, G., Babuška, R.: Drop-on-demand inkjet printhead performance improvement using robust feedforward control. In: Proceedings of the 50th IEEE Conference on Decision and Control (CDC-2011), pp. 4183–4188 (2011)

[64] Khalate, A.A., Bombois, X., Babuška, R.: Improving the performance of a drop-on-demand inkjet printhead. In: Proceedings of the 28th Benelux Meeting on Systems and Control, p. 104 (2009)

[65] Khalate, A.A., Bombois, X., Babuška, R., Scorletti, G., Koekebakker, S., Wijshoff, H., de Zeeuw, W., Waarsing, R.: Performance improvement of a drop-on-demand inkjet printhead: a feedforward control based approach. In: Proceedings of the 27th IS&T International Conference on Digital Printing Technologies (NIP27), pp. 74–78 (2011)

[66] Khalate, A.A., Bombois, X., Babuška, R., Waarsing, R., de Zeeuw, W., Klerken, P.: Robust feedforward control for a DoD inkjet printhead. In: Proceedings of the 29th Benelux Meeting on Systems and Control, p. 125 (2010)

[67] Khalate, A.A., Bombois, X., Babuška, R., Wijshoff, H., Waarsing, R.: Optimization-based feedforward control for a drop-on-demand inkjet printhead. In: Proceedings of the 2010 American Control Conference (ACC 2010), pp. 2182–2187 (2010)

[68] Khalate, A.A., Bombois, X., Babuška, R., Wijshoff, H., Waarsing, R.: Performance improvement of a drop-on-demand inkjet printhead using an optimization-based feedforward control method. Contr. Eng. Pract. **19**, 771–781 (2011)

[69] Khalate, A.A., Bombois, X., Scorletti, G., Babuška, R., Koekebakker, S., de Zeeuw, W.: Robust feedforward control for a drop-on-demand inkjet printhead. IEEE/ASME J. Microelectromechanical Syst. **21**, 1365–1374 (2012)

[70] Khalate, A.A., Bombois, X., Scorletti, G., Babuška, R., Waarsing, R., de Zeeuw, W.: Robust feedforward control for a drop-on-demand inkjet printhead. In: Proceedings of the 18th IFAC World Congress (IFAC 2011), pp. 5795–5800 (2011)

[71] Khalate, A.A., Bombois, X., Tóth, R., Babuška, R.: Optimal experimental design for LPV identification using a local approach. In: Proceedings of the 15th IFAC Symposium on System Identification (SYSID 2009), pp. 162–167 (2009)

[72] Khalate, A.A., Bombois, X., Ye, S., Babuška, R., Koekebakker, S.: Minimization of cross-talk in a piezo inkjet printhead based on system identification and feedforward control. J. Micromech. Microeng. **22**, Paper 115035 (2012)

[73] Komoto, H., Tomiyama, T.: Computational support for system architecting. In: Proceedings of the ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2010), Paper no. DETC2010-28683, pp. 25–34 (2010)

[74] Komoto, H., Tomiyama, T.: A system architecting tool for mechatronic systems design. CIRP Ann. - Manuf. Technol. **59**, 171–174 (2010)

[75] Komoto, H., Tomiyama, T.: Multi-disciplinary system decomposition of complex mechatronics systems. CIRP Ann. - Manuf. Technol. **60**, 191–194 (2011)

[76] Komoto, H., Tomiyama, T.: A theory of decomposition in system architecting. In: Proceedings of the 18th International Conference on Engineering Design (ICED11), vol. 2, pp. 334–343 (2011)

[77] Komoto, H., Tomiyama, T.: A framework for computer-aided conceptual design and its application to system architecting of mechatronics products. Comput.-Aided Des. **44**, 931–946 (2012)

[78] Kumar, A.: Adding schedulability analysis to the Octopus toolset. Master's thesis, Eindhoven University of Technology, Faculty of Mathematics and Computer Science, Design and Analysis of Systems group, Eindhoven (2011)

[79] Kwisthout, J.: Most probable explanations in Bayesian networks: complexity and tractability. Tech. Rep. ICIS-R10001, Radboud University Nijmegen, Institute for Computing and Information Sciences, Nijmegen (2010)

[80] Kwisthout, J.: Most probable explanations in Bayesian networks: complexity and tractability. Int. J. Approx. Reason. **52**, 1452–1469 (2010)

[81] Kwisthout, J.: Two new notions of abduction in Bayesian networks. Tech. Rep. ICIS-R10005, Radboud University Nijmegen, Institute for Computing and Information Sciences, Nijmegen (2010)

[82] Kwisthout, J.: Two new notions of abduction in Bayesian networks. In: Proceedings of the 22nd Benelux Conference on Artificial Intelligence (BNAIC 2010), pp. 82–89 (2010)

[83] Kwisthout, J.: The computational complexity of probabilistic inference. Tech. Rep. ICIS-R11003, Radboud University Nijmegen, Institute for Computing and Information Sciences, Nijmegen (2011)

[84] Kwisthout, J., Lucas, P.: Reasoning with different time granularities in industrial applications: A case study using CP-logic. In: Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC 2011), pp. 120–127 (2011)

[85] Kwisthout, J., Wareham, T., van Rooij, I.: Bayesian intractability is not an ailment that approximation can cure. Cogn. Sci. **35**, 779–784 (2011)

[86] Kwisthout, J.H.P., Bodlaender, H.L., van der Gaag, L.C.: The necessity of bounded treewidth for efficient inference in Bayesian networks. In: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010), pp. 237–242 (2010)

[87] Kwisthout, J.H.P., Bodlaender, H.L., van der Gaag, L.C.: The complexity of finding *k*th most probable explanations in probabilistic networks. In: I. Cerná, T. Gyimóthy, J. Hromkovic, K. Jefferey, R. Královic, M. Vukolic, S. Wolf (eds.) SOFSEM 2011: Theory and Practice of Computer Science, *Lecture Notes in Computer Science*, vol. 6543, pp. 356–367. Springer, Heidelberg (2011)

[88] Lensen, B.J.P.: Automated design-space exploration of resource-aware synchronous dataflow graphs using multi-objective evolutionary algorithms. Master's thesis, Eindhoven University of Technology, Faculty of Electrical Engineering, Eindhoven (2010)

[89] Moily, A.: Supporting design-space exploration with synchronous data flow graphs in the Octopus toolset. Master's thesis, Eindhoven University of Technology, Faculty of Mathematics and Computer Science, Software Engineering and Technology group, Eindhoven (2011)

[90] Schindler, K.: A simulator for data processing pipelines. Tech. rep., Stan Ackermans Institute / Software Technology, Eindhoven (2011)

[91] Sözer, H., de Roo, A., Akşit, M.: Architectural framework for energy optimization in embedded systems. In: Proceedings of the 2nd International Workshop on Software Research and Climate Change (WSRCC-2) (2010)

[92] Teeselink, E.: DPATCH: A tool chain for modeling and analyzing printer data path architectures. Tech. rep., Stan Ackermans Institute / Software Technology, Eindhoven (2010)

[93] Teeselink, E., Somers, L., Basten, T., Trčka, N., Hendriks, M.: A visual language for modeling and analyzing printer data path architectures. In: Proceedings of the Industry Track of Software Language Engineering 2011 (ITSLE 2011), pp. 1–20 (2011)

[94] Tomiyama, T.: Architecture-centric model-based product development. In: Proceedings of the 13th Mechatronics Forum International Conference (2012)

[95] Trčka, N., Hendriks, M., Basten, T., Geilen, M., Somers, L.: Integrated model-driven design-space exploration for embedded systems. In: Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XI), pp. 339–346 (2011)

[96] Trčka, N., Voorhoeve, M., Basten, T.: Parameterized timed partial orders with resources: Formal definition and semantics. ES Report ESR-2010-01, Eindhoven University of Technology, Department of Electrical Engineering, Electronic Systems group, Eindhoven (2010)

[97] Trčka, N., Voorhoeve, M., Basten, T.: Parameterized partial orders for modeling embedded system use cases: Formal definition and translation to coloured Petri nets. In: Proceedings of the 11th International Conference on Application of Concurrency to System Design (ACSD 2011), pp. 13–18 (2011)

[98] van der Laan, E., Cochior, C., Driessen, D.: A predictive control strategy for productive printers. In: Proceedings of the First DSPE Conference on Precision Mechatronics (DSPE-Conference 2012) (2012)

[99] van Rooij, I., Kwisthout, J., Blokpoel, M., Szymanik, J., Wareham, T., Toni, I.: Communicating intentions: Computationally easy or difficult? Front. Hum. Neurosci. **5**, 1–18 (2011)

[100] van Zuidam, S.: A scenario editor for design space exploration. Master's thesis, Eindhoven University of Technology, Faculty of Mathematics and Computer Science, Eindhoven (2009)

[101] Waarsing, B.J.W., van den Bosch, P.F.A., Hommersom, A.J.: Reprographic system. Patent Application PCT/EP2009/066348 (2010)

[102] Wareham, T., Kwisthout, J., Haselager, P., van Rooij, I.: Ignorance is bliss: A complexity perspective on adapting reactive architectures. In: Proceedings of the 2011 IEEE Conference on Development and Learning (ICDL 2011), pp. 465–470 (2011)

[103] Yang, Y.: Exploring resource/performance trade-offs for streaming applications on embedded multiprocessors. Ph.D. thesis, Eindhoven University of Technology, Eindhoven (2012)

[104] Yang, Y., Geilen, M., Basten, T., Stuijk, S., Corporaal, H.: Exploring trade-offs between performance and resource requirements for synchronous dataflow graphs. In: Proceedings of the 7th IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia 2009), pp. 96–105 (2009)

[105] Yang, Y., Geilen, M., Basten, T., Stuijk, S., Corporaal, H.: Automated bottleneck-driven design-space exploration of media processing systems. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2010), pp. 1041–1046 (2010)

[106] Yang, Y., Geilen, M., Basten, T., Stuijk, S., Corporaal, H.: Iteration-based trade-off analysis of resource-aware SDF. In: Proceedings of the 14th Euromicro Conference on Digital System Design (DSD 2011), pp. 567–574 (2011)

[107] Yang, Y., Geilen, M., Basten, T., Stuijk, S., Corporaal, H.: Playing games with scenario- and resource-aware SDF graphs through policy iteration. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2012), pp. 194–199 (2012)

[108] Ye, S.: Identification and feedforward control of a drop-on-demand inkjet printhead. Master's thesis, Delft University of Technology, Faculty of Mechanical, Maritime and Materials Engineering, Delft Center for Systems and Control, Delft (2011)

Index