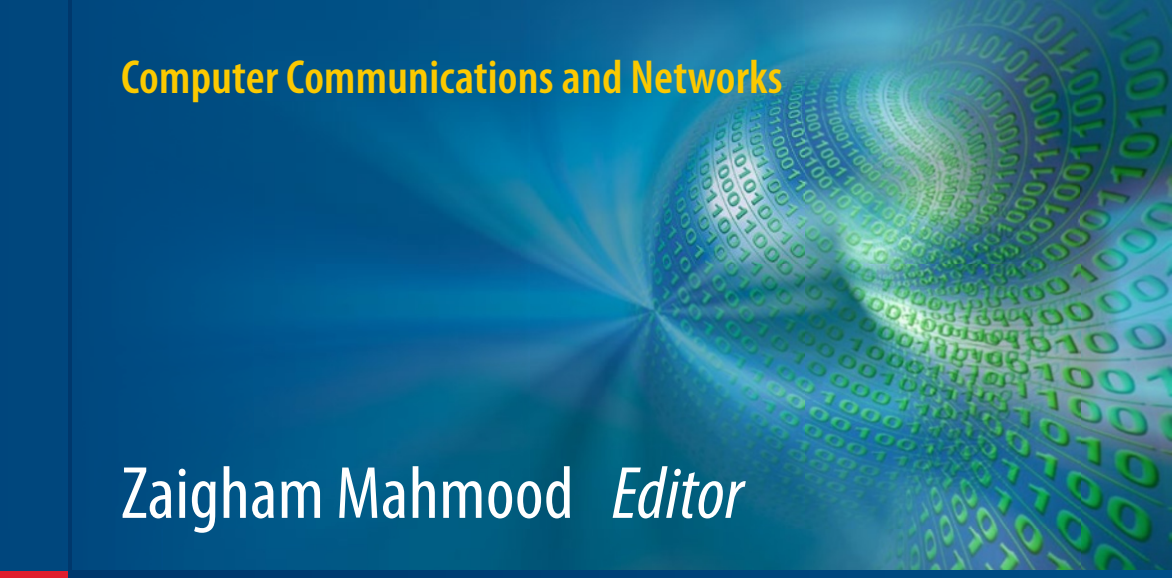
Computer Communications and Networks



Zaigham Mahmood *Editor*

Software Project Management for Distributed Computing

Life-Cycle Methods for Developing Scalable and Reliable Tools

Springer

Computer Communications and Networks

Series editor
A.J. Sammes
Centre for Forensic Computing
Cranfield University, Shrivenham Campus
Swindon, UK

The **Computer Communications and Networks** series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers, and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

More information about this series at http://www.springer.com/series/4198

Zaigham Mahmood
Editor

Software Project Management for Distributed Computing

Life-Cycle Methods for Developing Scalable and Reliable Tools

Springer

Editor
Zaigham Mahmood
Department of Computing and Mathematics
University of Derby
Derby, UK

Shijiazhuang Tiedao University
Hebei, China

ISSN 1617-7975 ISSN 2197-8433 (electronic)
Computer Communications and Networks
ISBN 978-3-319-54324-6 ISBN 978-3-319-54325-3 (eBook)
DOI 10.1007/978-3-319-54325-3

Library of Congress Control Number: 2017935573

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To
Zayb-un-Nisa Khan
On Her First Birthday,

With Best Wishes and
Prayers for a Long,
Contented and Happy Life

Preface

Overview

Software Project Management (SPM) is a subdiscipline of project management in which software projects are planned, engineered, implemented and monitored. In order that software products are delivered on time, within the allocated budgets and fully conforming to the user requirements, it is imperative that software projects are appropriately designed, managed and well executed, especially if these relate to complex software systems. Many SPM paradigms have been suggested in the past and successfully employed in recent years; however, with the advancement in computing technologies, including cloud computing, distributed computing and the Internet of Things, the existing frameworks and management approaches do not necessarily satisfactorily apply. There is now a requirement for software to be scalable, sustainable and suitable for distributed computing environments. This, in turn, suggests a requirement for management methods with evolutionary life cycles and software engineering approaches that take into account distributed working practices and distributed team management working in virtual operating environments. It is for this reason that SPM is becoming an important research topic in the field of software engineering.

In recent years, numerous newer approaches and tools for the development and management of software products and projects have been suggested, some being deployed with some degree of success to satisfy the requirements of scalability and multi-tenancy. However, the research must continue to fully satisfy the requirements as briefly mentioned above.

With this background, the current volume, *Software Project Management for Distributed Computing: Life-Cycle Methods for Developing Scalable and Reliable Tools*, aims to investigate the latest management approaches to developing complex software that is efficient, scalable, sustainable and suitable for distributed environments. The focus is primarily on newer methodologies with respect to management processes. Emphasis is also on the use of latest software technologies and

frameworks for the life-cycle methods including design, implementation and testing stages of the software development.

Hopefully, this text will fill a gap in the SPM literature and practice by providing scientific contributions from researchers and practitioners of international repute in the fields of management and software engineering. Thirty-six authors have presented latest research developments, frameworks and methodologies, current trends, state-of-the-art reports, case studies and suggestions for further understanding, development and enhancement of management approaches for developing scalable and multi-tenant complex software.

Objectives

The aim of this volume is to present and discuss the state of the art in terms of frameworks and methodologies for software project management for distributed computing environments. The features that set this book apart from others in the field include:

- Latest research, development and future directions in the proposed subject area of software project management (SPM)
- Case studies describing challenges, best practices and solutions for SPM for distributed computing environments
- Textbook and complete reference for students, researchers, practitioners and project managers in the subject area of SPM
- Alignment of software engineering frameworks with SPM approaches using latest technologies
- Corporate analysis presenting a balanced view discussing benefits and inherent issues

Organization

There are 15 chapters in *Software Project Management for Distributed Computing: Life-Cycle Methods for Developing Scalable and Reliable Tools*. These are organized in three parts as follows:

- Part I: *Characteristics and Estimation of Software Projects for Distributed Computing*. This section has a focus on characteristics, perspectives and estimation approaches. There are five chapters in this part of the book. The first two chapters discuss the modelling of reusability and estimation of cost and time for distributed software development projects. The focus of the third contribution in the section is on functional size measurement of distributed software applications, and the fourth chapter discusses the core characteristics of large-scale defence-related software projects. The fifth contribution

introduces and presents software project management (SPM) as a distributed service: SPMaaS.

- Part II: *Approaches and Frameworks for Software Development and Software Project Management*. This part of the book also comprises five chapters that focus on frameworks and methodologies. The first contribution presents component-based reference architecture for embedded software development. The next chapter proposes a 3PR framework for SPM based on people, processes, products and risks, while the third contribution discusses a novel crowdsourcing approach for software development. The fourth contribution presents a migration and management approach for distributed environments. The fifth chapter investigates a novel approach for modelling of large-scale multi-agent software systems.
- Part III: *Advances in Software Project Management and Distributed Software Development*. There are five chapters in this section as well that focus on latest developments in SPM and software development for distributed computing. The first contribution discusses an error proneness mechanism based on bird mating algorithm. The next chapter presents a novel Scrum process relevant to defence and security domain. The third chapter is on ontology annotation for SPM for distributed computing environments. The fourth contribution investigates the scope of Agile project management in an educational setting, while the final chapter in the book focusses on SPM for combined software and data engineering.

Target Audiences

The current volume is a reference text aimed at supporting a number of potential audiences, including the following:

- *Project managers and software engineers* who wish to deploy the newer approaches and technologies to ensure the development of software that is scalable, sustainable and suitable for distributed computing environments
- *Students and lecturers* who have an interest in further enhancing the knowledge of technologies, mechanisms and frameworks relevant to software project management (SPM) from a distributed computing perspective
- *Researchers* in this field who require up-to-date knowledge of the current practices, mechanisms and frameworks relevant to SPM, to further extend the body of knowledge in this field

Derby, UK
Hebei, China

Zaigham Mahmood

Acknowledgements

The editor acknowledges the help and support of the following colleagues during the review, development and editing phases of this text:

- Josip Lorincz, FESB-Split, University of Split, Croatia
- Dr. N. Maheswari, School CS & Eng, Chennai, Tamil Nadu, India
- Aleksandar Milić, University of Belgrade, Serbia
- Dr. S. Parthasarathy, Thiagarajar College of Eng, Tamil Nadu, India
- Daniel Pop, Institute e-Austria Timisoara, West Univ. of Timisoara, Romania
- Dr. Pethuru Raj, IBM Cloud Center of Excellence, Bangalore, India
- Dr. Muthu Ramachandran, Leeds Beckett University, Leeds, UK
- Dr. Lucio Agostinho Rocha, State University of Campinas, Brazil
- Dr. Saqib Saeed, University of Dammam, Saudi Arabia
- Dr. Mahmood Shah, University of Central Lancashire, Preston, UK
- Dr. Fareeha Zafar, GC University, Lahore, Pakistan

I would also like to thank the contributors of this book: 36 authors and co-authors, from academia as well as industry from around the world, who collectively submitted 15 chapters. Without their efforts in developing quality contributions, conforming to the guidelines and meeting often the strict deadlines, this text would not have been possible.

Grateful thanks are also due to the members of my family – Rehana, Zoya, Imran, Hanya, Arif and Ozair – for their continued support and encouragement. Every good wish, also, for the youngest in our family: Eyaad Imran Rashid Khan and Zayb-un-Nisa Khan.

Department of Computing and Mathematics
University of Derby
Derby, UK

Zaigham Mahmood

Shijiazhuang Tiedao University
Hebei, China

Other Springer Books by Zaigham Mahmood

Data Science and Big Data Computing: Frameworks and Methodologies

This reference text has a focus on data science and provides practical guidance on big data analytics. Expert perspectives are provided by an authoritative collection of 36 researchers and practitioners, discussing latest developments and emerging trends, presenting frameworks and innovative methodologies and suggesting best practices for efficient and effective data analytics. ISBN: 978-3-319-31859-2

Connectivity Frameworks for Smart Devices: The Internet of Things from a Distributed Computing Perspective

This is an authoritative reference that focuses on the latest developments on the Internet of Things. It presents state of the art on the current advances in the connectivity of diverse devices and focuses on the communication, security, privacy, access control and authentication aspects of the device connectivity in distributed environments. ISBN: 978-3-319-33122-5

Cloud Computing: Challenges, Limitations and R&D Solutions

This reference text reviews the challenging issues that present barriers to greater implementation of the cloud computing paradigm, together with the latest research into developing potential solutions. This book presents case studies and analysis of

the implications of the cloud paradigm from a diverse selection of researchers and practitioners of international repute. ISBN: 978-3-319-10529-1

Continued Rise of the Cloud: Advances and Trends in Cloud Computing

This reference volume presents latest research and trends in cloud-related technologies, infrastructure and architecture. Contributed by expert researchers and practitioners in the field, this book presents discussions on current advances and practical approaches including guidance and case studies on the provision of cloud-based services and frameworks. ISBN: 978-1-4471-6451-7

Cloud Computing: Methods and Practical Approaches

The benefits associated with cloud computing are enormous; yet the dynamic, virtualized and multi-tenant nature of the cloud environment presents many challenges. To help tackle these, this volume provides illuminating viewpoints and case studies to present current research and best practices on approaches and technologies for the emerging cloud paradigm. ISBN: 978-1-4471-5106-7

Software Engineering Frameworks for the Cloud Computing Paradigm

This is an authoritative reference that presents the latest research on software development approaches suitable for distributed computing environments. Contributed by researchers and practitioners of international repute, the book offers practical guidance on enterprise-wide software deployment in the cloud environment. Case studies are also presented. ISBN: 978-1-4471-5030-5

Cloud Computing for Enterprise Architectures

This reference text, aimed at system architects and business managers, examines the cloud paradigm from the perspective of enterprise architectures. It introduces fundamental concepts, discusses principles and explores frameworks for the adoption of cloud computing. The book explores the inherent challenges and presents future directions for further research. ISBN: 978-1-4471-2235-7

Requirements Engineering for Service and Cloud Computing

This text aims to present and discuss the state of the art in terms of methodologies, trends and future directions for requirements engineering for the service and cloud computing paradigm. Majority of the contributions in the book focus on requirements elicitation, requirements specifications, requirements classification and requirements validation and evaluation. ISBN: 978-3319513096

User Centric E-Government: Challenges and Opportunities

This text presents a citizens-focused approach to the development and implementation of electronic government. The focus is twofold: on challenges of service availability and e-service operability on diverse smart devices as well as on opportunities for the provision of open, responsive and transparent functioning of world governments. It is forthcoming.

Contents

Contributors

Tarek A. Ali Department of Computer Science, Institute of Statistical Studies and Research, Cairo University, Giza, Egypt

Luigi Benedicenti University of Regina, Regina, Canada

Vikrant Chaugule Department of Computer Science and Engineering, National Institute of Technology, Surathkal, Karnataka, India

Paolo Ciancarini University of Bologna, Bologna, Italy

Franco Cotugno Italian Army General Staff, Rome, Italy

Jim Davies Software Engineering Group, Department of Computer Science, University of Oxford, Oxford, UK

Kadir Alpaslan Demir Department of Software Development, Turkish Naval Research Center Command, Istanbul, Turkey

Manal El Bajta Software Project Management Research Team, ENSIAS, Mohammed V University, Rabat, Morocco

José Luis Fernandez-Aleman Software Engineering Research Group, Regional Campus of International Excellence, "Campus Mare Nostrum", University of Murcia, Murcia, Spain

Filomena Ferrucci University of Salerno, Fisciano, Italy

Mervat H. Gheith Department of Computer Science, Institute of Statistical Studies and Research, Cairo University, Giza, Egypt

Jeremy Gibbons Software Engineering Group, Department of Computer Science, University of Oxford, Oxford, UK

Carmine Gravino University of Salerno, Fisciano, Italy

Ali Idri Software Project Management Research Team, ENSIAS, Mohammed V University, Rabat, Morocco

Harsh Jain Department of Information Technology, Indian Institute of Information Technology, Allahabad, India

Manish Kumar Department of Information Technology, Indian Institute of Information Technology, Allahabad, India

P. Mangayarkarasi Visvesvaraya Technological University, Belgaum, India

Syed Hassan Marzooq Curtin University, Perth, WA, Australia

Angelo Messina Innopolis University, Innopolis, Russian Federation

Eman S. Nasr Independent Researcher, Cairo, Egypt

Joaquín Nicolas Ros Software Engineering Research Group, Regional Campus of International Excellence, "Campus Mare Nostrum", University of Murcia, Murcia, Spain

Bogdan Okreša Đurić Artificial Intelligence Laboratory, Faculty of Organization and Informatics, University of Zagreb, Varazdin, Croatia

Udsanee Pakdeetrakulwong Curtin University, Perth, WA, Australia

Amrit Pal Department of Information Technology, Indian Institute of Information Technology, Allahabad, India

Simon P Philbin Enterprise Division, Imperial College London, London, UK

Muthu Ramachandran School of Computing, Creative Technologies and Engineering, Leeds Beckett University, Leeds, UK

Pasquale Salza University of Salerno, Fisciano, Italy

Areeg Samir Faculty of Computer Science, Libera Università di Bolzano, Bolzano, Italy

R. Selvarani Department of CSE, ACED Alliance University, Bangalore, India

Seyyed M. Shah Software Engineering Group, Department of Computer Science, University of Oxford, Oxford, UK

Alberto Sillitti Innopolis University, Innopolis, Russian Federation

Giancarlo Succi Innopolis University, Innopolis, Russian Federation

Ambrosio Toval Software Engineering Research Group, Regional Campus of International Excellence, "Campus Mare Nostrum", University of Murcia, Murcia, Spain

James Welch Software Engineering Group, Department of Computer Science, University of Oxford, Oxford, UK

Pornpit Wongthongtham Curtin University, Perth, WA, Australia

Bo Xing Computational Intelligence, Robotics, and Cybernetics for Leveraging E-future (CIRCLE), Institute of Intelligent System, Faculty of Engineering and the Built Environment, University of Johannesburg, Johannesburg, Gauteng, South Africa

About the Editor

Professor Dr. Zaigham Mahmood is a published author of 19 books, 6 of which are dedicated to electronic government and the other 13 focus on the subjects of cloud computing, data science, big data, Internet of Things, project management and software engineering, including *Cloud Computing: Concepts, Technology and Architecture* which is also published in Korean and Chinese languages. Additionally, he is developing two new books to appear in 2018. He has also published more than 100 articles and book chapters and organized numerous conference tracks and workshops. Professor Mahmood is the editor-in-chief of the *Journal of E-Government Studies and Best Practices* as well as the series editor-in-chief of the IGI book series on *E-Government and Digital Divide*. He is a senior technology consultant at Debesis Education UK and associate lecturer (research) at the University of Derby, UK. He further holds positions as professor at Shijiazhuang Tiedao University in Hebei, China, and as foreign professor at NUST and IIU in Islamabad, Pakistan. Professor Mahmood is also a certified cloud computing instructor and a regular speaker at international conferences devoted to cloud computing and e-government. His specialized areas of research include distributed computing, project management and e-government.

Part I
Characteristics and Estimation of Software Projects for Distributed Computing

Chapter 1
Modeling of Reusability Estimation in Software Design with External Constraints

R. Selvarani and P. Mangayarkarasi

1.1 Introduction

With the increasing trends of demands in dynamic applications by the users, the software project development organizations are encountering a stiff competition with each other in order to cater up such demands. Although new programming tool along with new technologies evolved every 5–10 years of time, still the bases of all the applications are ultimately programs and codes. Hence, source code becomes a valuable asset for any IT firms, which is also considered as an intellectual property for its existing customers. Apart from this, usually the software system sometimes may not meet with consideration of the users due to some massive activities on the design and the accomplishment of the software. An important reason for some of the failures is the lack of ability to respond to the various needs of the user requirements in the industry. Although there are various software, frameworks, and tools that can perform a smart project management, there has always been a lesser degree of inclination of the software engineering to be considered in project management much. From the designers' perspective, they evaluate the design on reusability by how easily and how effectively it can be used by the customers in different types of working environments. High-quality design systems have become important to many academic researchers as well as commercial software builders. It is measured by the way of flexibility, understandability, and reusability [1]. Figure 1.1 highlights the conventional iterative process of design phase on reusability concept.

R. Selvarani
Department of CSE, ACED Alliance University, Bangalore, India
e-mail: selvarani.r@alliance.edu.in

P. Mangayarkarasi (✉)
Visvesvaraya Technological University, Belgaum, India
e-mail: mangaivtu@gmail.com

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_1

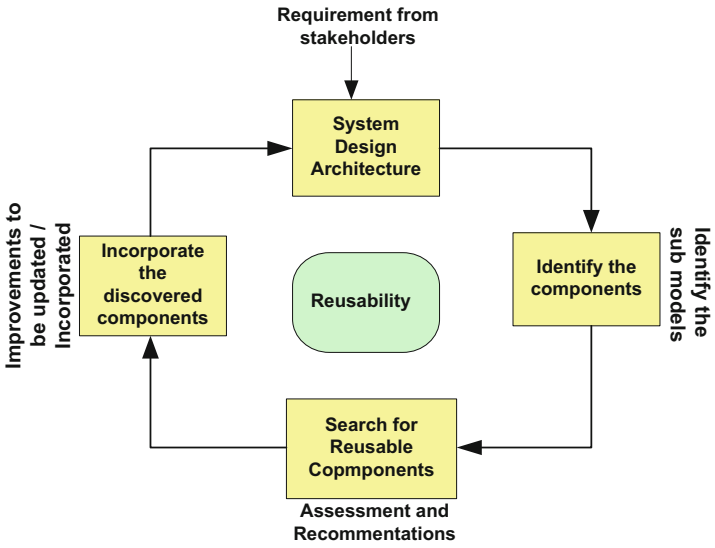


Fig. 1.1 Conventional iterative process of design phase

The concept of design reusability is meant for mitigating all the uncertainties in the software design and development process. Such reusability theory is originally meant for upgrading the software components by deploying existing ones. A health practice of design reusability will always assist in providing better quality, productivity, reliability, minimal expenditure, and less probability of schedule slippage. Design reusability is also dependent on the type of the projects that a team handles. Let us assume that a team of 15 members is handling mobile application projects from the past 7 years; then it is obvious that this team has increased their experience and skills in design and development of mobile-based application. Hence, there is a more likelihood that in future also they will be given similar kind of projects. However, it can be assumed that this team of 15 members will not be given a very new domain which they have zero work experience, but they may be given a project which is little advanced form of mobile-based application with certain degree of new components to be designed. In order to maintain a better turnaround time, it is necessary that the team should explore the new challenges by classifying their problems as – (i) find out what part of design components for new software project matches with their earlier deliveries and (ii) find out probability of cost involved in new development. Hence, there is always a preliminary investment in order to perform design reusability modeling, but the return of investment of such modeling is always good. Moreover, the development team starts working on exploring and designing reusable design components; an organization is creating a valuable base of design knowledge that has positive impact on productivity, financial planning, and risk assessment.

There are two types of reuse practices, viz., horizontal reuse and vertical reuse [2]. Horizontal reuse is mainly deployed for software components that are

incorporated on multiple types of applications. The system also consists of typical library components, e.g., routines for string manipulation, class of linked list, functions for user interface, etc. Various third party applications are also used in horizontal reuse practices, and it is one of the most frequently used practices in software engineering. Vertical reuse is quite less used but has a potential impact on the design aspects. The core idea of the vertical reuse is to reuse the area of system function as well as system domain. To nurture the practices of design reusability, the following are some factors to be considered:

- Clear-cut organization structure of software development: This will assist in proper archiving of the codes and IP (intellectual property) with all the protocols for security in accessing and usage structure.
- Higher analytical and problem-solving skill: The technical team should have higher problem-solving scheme to identify an appropriate segment of reusable design without violating the copyright of IP.
- Good exposure to risk assessment: The technical team should be well aware of the risk management and avoidance planning to avoid uncertainties of selecting and planning for development of the reusable design.

This chapter reviews problems about some of the existing techniques to reusability concept and introduces a novel framework that has the capability to precisely estimate design reusability along with optimization. The proposed model discusses discretely reusability estimation technique followed by optimization technique to perform cost-effective design reusability in software engineering. Section 1.2 discusses the related works associated with reusability concept in software engineering followed by a discussion on problem formulations in Sect. 1.3. The contribution of the proposed model for the reusable model is elaborated in Sect. 1.4 followed by research methodology in Sect. 1.5. The design principle is discussed in the following section, while the result discussion is carried out in Sect. 1.7. Finally, some concluding remarks are presented in Sect. 1.8.

1.2 Related Work

This section about the existing studies is carried out in the area of reusability about software engineering. Our prior study already reviewed existing techniques of enhancing design reusability in software engineering [3]. Recently, the work carried out by Alonso et al. [4] discussed the standard framework usage for reusability. Thakral et al. [5] presented a review work on reusability aspect pertaining to software development. A similar form of discussion toward challenges and importance of software reusability was presented by Soora [6]. Most recently, Tahir et al. [7] presented modeling of reusability emphasizing on components of software. Various techniques are briefed by the authors. Ahmaro et al. [8] discussed the exercised reusability of software methods considering case study of Malaysia using the qualitative approach. Singh and Singh [9] discussed reusability

framework considering the case study of cloud computing. The authors used a simulator called as CloudSim to validate their model. Basha and Moiz [10] discussed different techniques of reusability and introduced a methodology to perform configuration of vulnerable software components using coupling between object (CBO) metric. Zhu et al. [11] developed a framework that focuses on reusing a specific component in computational model.

Xiao et al. [12] developed a universal protocol for strengthening the reuse factor by considering multiple agent systems. The authors emphasized the usage of the design patterns for high-end architectures. The work carried out by Gupta and Rathi [13] introduced a modeling using predictive principle for forecasting the reusable software components based on original requirement of stakeholder and similarity match. The technique also used Rabin-Karp algorithm for carrying out feature selection, genetic algorithm for optimization, and k-means for clustering operation. Adoption of evolution technique was also witnessed in the recent work of Singh et al. [14] where the author adopted fuzzy logic for evaluating their reusability modeling. The study considers multiple valued logical inputs for cohesion, coupling, size, and complexity to compute reusability. Monga et al. [15] and Geertsema [16] studied the potential influence of multiple parameters on software reusability. The prime emphasis was laid over the quality parameters as well as selection of a criterion of the software attributes, e.g., maintainability, understandability, flexibility, quality, portability, complexity, cost, size, and independence. The study outcome was assessed using cyclomatic complexity and Halstead complexity.

Lani et al. [17] developed a technique to deal with the complex mathematical problems in the storage system. Nuseibeh et al. [18] studied about the inconsistencies in the software engineering. The problems pertaining to software reuse have been highlighted in the work of Andreou and Papatheocharous [19] where a framework has been presented for identifying precise software components. The framework targets to perform profiling that can directly assist in recognizing the actual requirements of the system to increase the reusability factor. The study also implements instances of ontologies to perform matching of software components. Bombonatti et al. [20] presented a study of associated software reusability with various nonfunctional software requirements along with a discussion of contextual attributes. This study has particularly pointed at the role played by human factor in an organization in software reusability concept. However, the entire study is based on quantitative approach and doesn't solve any problem. The contribution of this paper is its findings from the survey about the importance of many contextual factors. Similar direction of work was also carried out by Muller [21] where the authors emphasized software reusability with an aid of multiple factors, e.g., challenges in an organization, evolution, issues in integrations, the demand of reuse, etc. Rezaee and Pajohesh [22] have introduced a programmatic approach to discuss software reusability. Oliveira et al. [23] have addressed the problem of extraction of reusable assets in software in order to develop a recommendation-based modeling for reusing such components. Different methods and classes are considered as object-oriented entities that were extracted in this study to find out methods or classes with equivalent name. Sharma et al. [24] have presented a

component-based software framework in order to enhance the operationality of software reuse. The authors have presented an empirical-based approach that can select software reusable components using evolutionary algorithms.

Stefi et al. [25] have adopted contingency approach to study the reusability factor of external software components. Using qualitative-based study, the authors have gathered primary information from the participants about the positive influence of reusing external software components. The author's discussed modeling by the associate structure of decentralized organization, long-term orientation, and modularity in IT architecture with reusability of external software components. Subedha and Sridhar [26] have addressed the problem of the verifying model for extracting software components. The authors have also presented a contextual modeling approach for software reusability along with optimization performed on it. The prime contribution is task division for software reusability with respect to identification of components, extraction of component, and qualification of components. The study outcome was assessed using time required for extraction. Tibermacine et al. [27] have presented a framework in order to address the constraints of software architectures based on reusability concept.

There are many studies focused on enhancing reusability, but the trade-offs being explored are, viz., (i) very few studies are considered (Small and Medium Enterprises (SME)) into account while developing reusability factor, (ii) existing studies on reusability lack any forms of numerical benchmarking considering the frequently used software designs, (iii) software metric suites are in infancy stage with respect to real-time possible constraints, and (iv) there is lack of computational and optimization model in developing software reusability. Therefore, the absence of all the above points renders to discuss the problems by considering the real-time constraints for SMEs in software engineering. The next section discusses the problems being formulated for the proposed model.

1.3 Problem Formulation

The proposed study considers the problem as – how to ensure better design reusability in object-oriented software project development? The problem is particularly focused on small and medium enterprise who has small number of human resource with more number of tasks to do from multiple clients. Hence, this is the best possible scenario to check if an efficient optimization policy can be laid by a novel technique. The proposed system identifies three different constraints for such organization, viz., human resource, cost, and work schedule. Based on these constraints, the problem is how to optimize productivity retaining the same or highly minimal resources. The problem pertaining human resource is basically to explore the possibility of using similar resources for ensuring increased productivity. Similarly, the problem pertaining to cost will focus on exploring the technique of minimizing operational cost of production without affecting the quality of delivery. Finally, work schedule is a hard constraint which will explore the chances

of increased productivity keeping the uniform working schedule. The term productivity is strongly associated with design reusability, which will mean if the value of design reusability of software projects is more, it means productivity (or quality of the product) has been upgrading.

In order to understand the problem scenario, let us consider the criticality of the considered constraints for the proposed system with respect to optimization:

- *Human resource*: Smaller organization has smaller number of human resources who are always endowed with unproportionate workload. Although there is always a breakpoint for the capability of the human resources, it is not possible to identify the variable capacity of multiple human resources. For example, for a given same job, one employee can have more efficiency as others. Hence, modeling human resource as a constraint is one of the challenging tasks if considered along with design reusability concept.
- *Work schedule*: There is always constant work timing for all the human resources. However, the challenge is how much of the same work schedule can be used by an employee to design a new component with maximum design reusability? There may be multiple unseen factors that may have positive or negative influence on the employee's productivity (in perspective of designing reusable components) in static work schedule.
- *Cost*: The term cost will mean amount of resources required by an employee to develop a new software project with maximum reusable components. For better margin of profit, the cost of new development must be as low as possible. Although cost cannot be lowered down to zero, it is challenging to perform predictive modeling to foresee some approximated value of cost of new development of upcoming software project.

Hence, all the three real-life constraints, e.g., human resources, work schedules, and cost, are most associated with uncertainty factors which are required to be computed using optimization principle. More elaboration of the problem is given below:

To understand problem considered, we consider three phases of proposed system, i.e., requirement gathering, evaluating existing design attributes, and analyzing tentative cost of new development. Figure 1.2 highlights three essential blocks, i.e., (i) requirement block, (ii) existing design attribute block, and (iii) new design block. The requirement block represents a process management step that reviews new project requirement of clients with a special insight on its design attributes. The existing design attribute block performs the check if the new design could be compatibly mapped with any existing reusable components with certain thresholds. It evaluates the size of reusable components to be used in new product design. The new design block basically evaluates the exact amount of reusable design for the new segment of the product development from the scratch. The prime challenge in this block is to perform a check if the newly developed content does reach some specific limit of design reusability standard imposed by an organization. Not only this, performing optimization principle over software engineering was less attempted in prior research work that will also pose a serious impediment toward

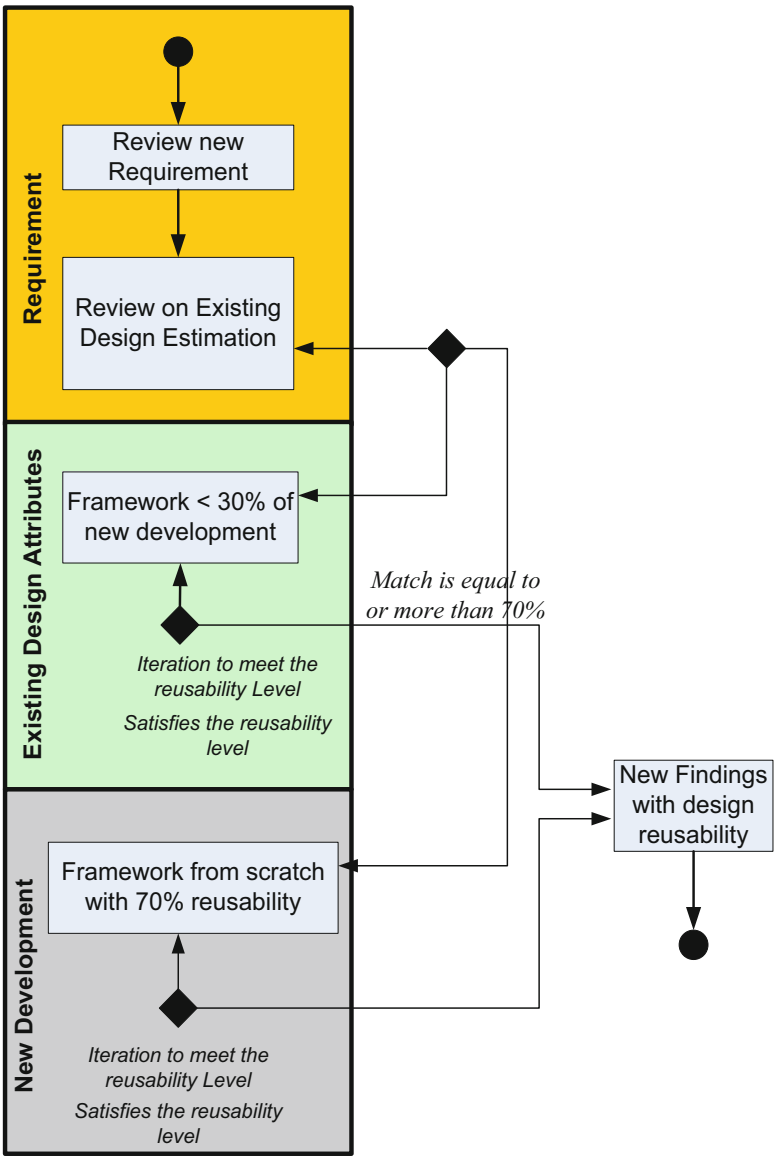


Fig. 1.2 Schematic representation of problem formulations

evolving up a new model for design reusability. The next section is about the proposed system and its significant contribution to address the problem.

1.4 Proposed Modeling

The prime purpose of the proposed system is to introduce a novel modeling approach of optimization for estimating design reusability for software project. The proposed system considers the real-time constraints and presents a unique technique to enhance project management in software development firms. The development scenario considers the case study of small and medium enterprises which adheres to meet time- and mission-critical objection using small amount of resources. From software engineering viewpoint, such corporates have higher inclinations toward adoption of cost-cutting measures in production. Hence, the proposed model offers a technique which is essentially meant for the stakeholder of small and medium enterprises for evaluating two facts, viz.:

- How much reusable design components can be used for meeting new software project development?
- Can the new production cost be lowered by incorporating reusable design components in it?

The prime contributions of the proposed modeling approach are as follows:

- *Novel approach in software engineering*: For the first time in research, the proposed model considers the real-world constraints for evolving up with a mathematical model for enhancing the project management methodologies. The proposed technique is designed based on simple tool, and modeling supports extensive mathematical operations to superiorly enhance the optimization process.
- *Efficient design reusability*: The proposed model uses probability theory enriched with statistical operation in order to perform modeling. The technique uses simple parameters like number of days, human resources, cost of the production, working schedule, etc. and modeled them in simple design reusability estimation technique.
- *Highly adoptable design principle*: The development of the proposed technique uses open-source platform for extracting the software metric suite information (CK metrics). Hence, the design principle can be considered to be platform independent and can offer better flexibility of adoption even if different domains of problems are given.
- *Robust optimization technique*: The proposed model uses backpropagation learning technique which is used to ensure superior consistency and reduced computational complexity in reusability estimation. It also uses Levenberg-Marquardt optimization technique, which is used for solving nonlinear optimization problems. The problems considered in our case are also nonlinear in nature; hence it finds better suitability.

The architecture of the proposed modeling is highlighted in Fig. 1.3. The architecture is meant for taking four different inputs (number of developer, number of client, cost, and errors) in order to perform estimation and optimization of amount

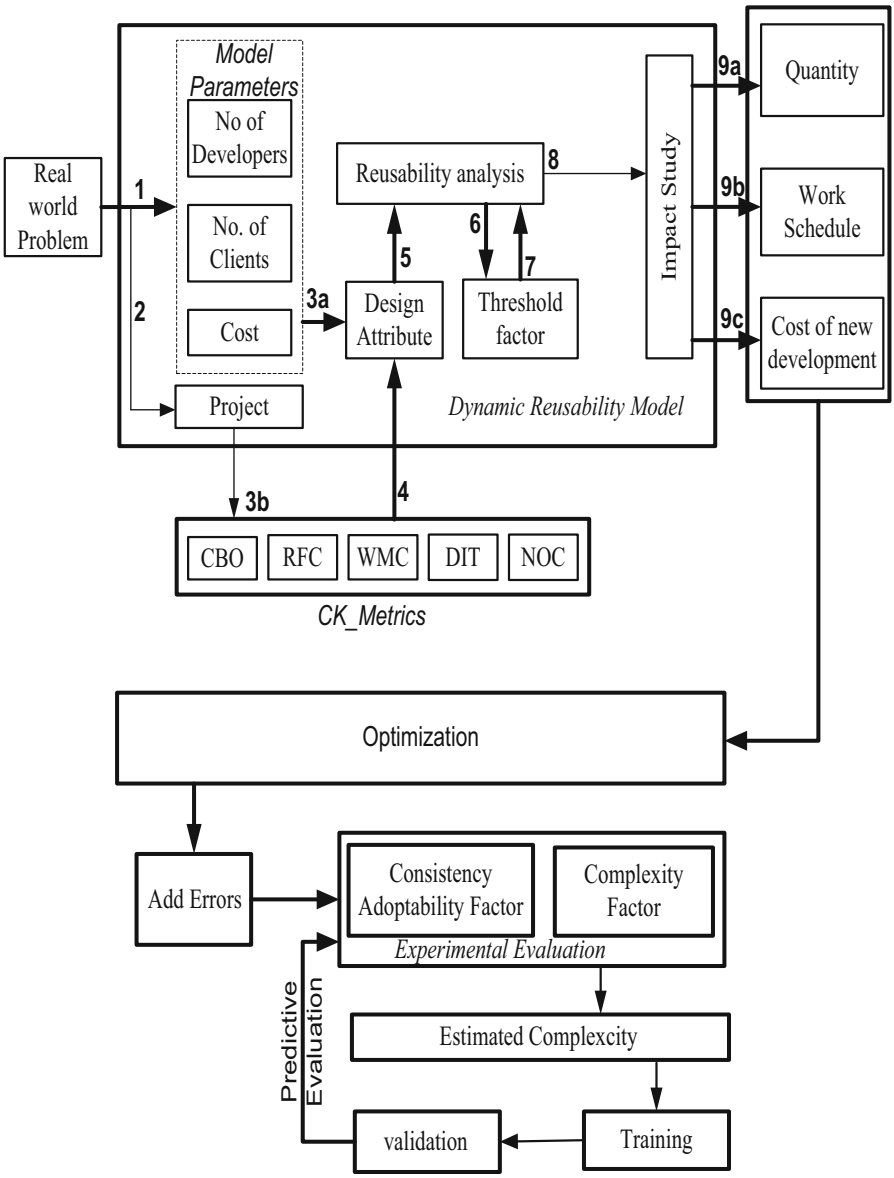


Fig. 1.3 Proposed modeling of optimized reusability estimation

of reusable design components to be used. The proposed technique is essentially meant for minimizing the new production cost along with effort and time without any forms of negative influence to the quality of software project delivery.

1.5 Research Methodologies

The design of the proposed model is carried out using analytical research approach emphasizing the critical factors involved in software development projects. The flow of the adopted research methodology is highlighted in Fig. 1.4. The brief discussions of an essential building block of the adopted research methodology are as follows:

- *Data collection*: We consider some sample software projects (supply chain management) which were developed using JSP, Servlets, or Struts just in order to ensure that we have a good number of logical codes for developing designs. We use Metrics 1.3.6 [28] tool in order to extract the original values of CK metrics, e.g., CBO, RFC, WMC, DIT, and NOC.
- *Assumption building*: The study considers the real-life scenario of SMEs where resources are always subjected to optimization. The three problems considered are (i) resource quantity, (ii) work schedules, and (iii) cost of new development. The proposed optimization technique is designed considering hard thresholding scheme based on these three constraints.
- *Descriptive statistics*: The numerical outcome of the study uses simple descriptive statistics (mean, median, mode, skewness, kurtosis, variance, standard deviation, etc.) to observe the trend of data.
- *Inferential Statistics*: The study uses t-test and analysis of variance in order to arrive at the p value of the numerical data. This is basically used for computing significance value for all the CK metrics in order to check its possible impact on design reusability.
- *Formulate condition of reusability*: We consider that formulation of code reusability should be made in highly flexible way. We assume that a developer has a possession of software design for a delivered project. Assuming the organization is working on the similar domain, the developer can be easily thought of getting a similar kind of software project with minor or some amount of changes. Hence, the proposed model will be used to identify what amount of design for new project matches with existing reusable design components (from the prior project). The system will also need to design reusable components whose cost will need to be determined. Hence, we assume a fixed threshold point where the new component to be developed should have a minimum amount of specific design reusability.
- *Consider error*: Uncertainty is always a part of project management as well as in software development life cycle. Hence, we consider adding up hypothetical errors to check the fault tolerance of the proposed model for any future upcoming risk matching with incorporated errors.
- *Apply BPA*: The study uses backpropagation algorithm (BPA) to perform optimization [29]. The reason of adoption of this algorithm is because (i) it is the most simple optimization technique and (ii) it has faster convergence on required local minima.

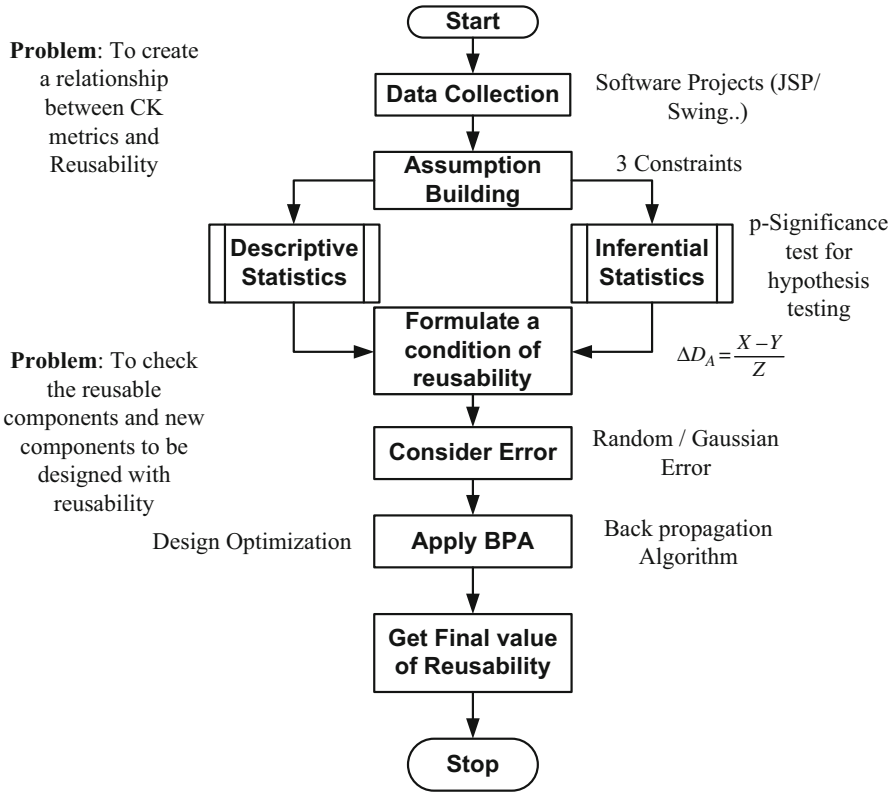


Fig. 1.4 Flow of the research methodology adopted

- *Get final value of reusability*: The final judgment of the proposed technique is carried out considering the evaluation of consistency factor and complexity factor. As the study uses a statistical approach, hence, inferences of the accomplished outcomes are quite comprehensive.

1.6 Design Principle

In our proposed approach, the project is broken down into two analysis groups, viz., reusability analysis and controlling analysis. The analysis is forward looking and tells us what needs to be reused, where it is to be done, how it is to be reused, and who is going to do it. Analysis usually occurs prior to embarking upon a project or early in the project life cycle. Controlling is intended to identifying and developing the new requirements which is deviated from the existing plan. It takes more narrow

and immediate focus with the intention for alerting managers to significantly deviate from plan while the project is in the process, where it is mandatory to allot the work to senior developers, who already work in the similar project. The proposed model considers multiple research approaches in order to fulfill the research goal. A brief discussion of the objectives is carried out based on two core module developments as follows:

1.6.1 Design of Reusability Estimation

The real-time software development methodologies are potentially affected by various extrinsic and intrinsic factors. Such factors have a direct impact on the reusability phenomenon. Hence, we consider some of the real-time constraints which are witnessed in almost every project team. The study considers three such real-time constraints, e.g., (i) human resources, (ii) cost, and (iii) work schedule. The supporting justifications are briefed below:

- *Human resources*: Human resource will directly mean project team members who are technically skilled to develop and deliver software projects:
  - Justification: This factor plays a crucial role in our proposed model as our investigation is focused toward exploring optimal number of team members required to deliver a task which is bigger in dimension with higher scale of difficulty in tentative number of workdays. The study will investigate the fact if lower number of human resource can still deliver the quality in software projects keeping profit in mind in terms of design reusability.
  - Feature: The prime feature of this module will be to develop new applications by reusing preexisting designs. Consequently development effort might be low when the designs are predominantly reused.
  - Correlation: Cost is measured as development effort which is directly proportional to design reusability.
- *Cost*: The factor of cost can be defined as total expenditure borne by the project team to deliver a software project with higher degree of design reusability:
  - Justification: Consider an example that for a given software project design, only 40% of the design components stored in database of projects could be reused for new projects. Hence, a question arise that 60% of new development has to be carried out for new projects where it is quite challenging part to understand how much part of these 60% new design could be reusable. Hence, a threshold-based factor is used which will always ensure that a software development must propose a new design in such a way that it must not be below the threshold limit in order to keep the cost as low as possible.

  - Feature: Cost calculation will be done on the basis on number of hours involved, inclusion of new resources, score of new reusable design, etc. Hence the capital investment on new development is essential.
  - Correlation: Capital cost is measured as scratch development effort, which is directly proportional to new design with respect to reusability in future needs.
- *Work schedule*: Work schedule will directly mean working hours:
  - Justification: Smart employee management and performance enhancement schemes always look for enhancement in production along with skill adhering to same working hours. However, working hours are also governed by various facts, e.g., how much skillful and target oriented an employee is or inefficiency of team leader to make employee work off the working time. Or it may be skill gap even. Ineffective work schedule results in inferior quality and degraded productivity.
  - Feature: The module of work schedule will look for the capability of an employee to hold multiple responsibilities of multiple clients on stipulated period of time.
  - Correlation: Reuse of design considering invariants of work schedules for multiple clients by a single developer. It is measured as rates to arrive at the time and effort needed to complete the project.

The framework considers the formulations of real-world problem, i.e., human resources, cost, and work schedule. Initially, the study investigates software projects designed by open-source programs, e.g., Java. We have used different types of available tools, e.g., Metrics 1.3.6, for obtaining values in CK metrics. The framework also extracts CK metrics from the object-oriented software projects, i.e., CBO, RFC, WMC, DIT, and NOC. Based on the input, let $D_A \in$ CK={CBO, RFC, WMC, DIT, NOC } be one of the feature attributes where, initially, it can be possible to find mean of their daily changes for each candidate class in object-oriented programs over the whole development period not including days when the class has been reused. Therefore,

$$X = \sum_{k=1}^{N} D_A(k.\Delta e) \tag{1.1}$$

and

$$Y = \sum_{k=1}^{N} D_A((k-1).\Delta e) \tag{1.2}$$

$$Z = N - |T| \tag{1.3}$$

where N is total number of days required to develop the complete projects ($P_{\max} = P_1 + P_2 + \ldots P_n$) and Δe is time required in one single day effort. It should

be noted that T is the actual duration of development required by the developers for performing design reusability and $T<N$. This mean value for feature attribute CK is represented as

$$\Delta D_A = \frac{X - Y}{Z} \tag{1.4}$$

The above equation yields the mean of the daily changes of quality CK metrics only for the days in which a class has been reused. Therefore, the proposed model only emphasizes on the days which were used for making the reusable components. A closer look into all the above equation will show that if D_A is lowered down, then ΔD_A will be considered to be optimized. It will mean that although CK metrics gives potential information about the underlying association of various design dependencies, higher values of any CK metrics are detrimental for an efficient design principle. Hence, the proposed model can easily use Eq. 1.4 to assess if their formulated design has better reusability or not. The complete approach is based on non-iterative optimization principle which is faster and simple for the stakeholder to assess their tentative profit margin by extracting maximum value of design reusability. It also assists them to formulate strategical planning for future software design requirements in order to ensure maximum reusable components, which are very important in software engineering.

1.6.2 Design of Optimization in Reusability Estimation

The previous module of the study assists in calculating design attribute in terms of reusability. This module of the study is more focused on performing optimization. Therefore, along with human resource, cost, and work schedule, a new attribute called as an error is also considered. Hence, the study performs optimization using backpropagation algorithm on the four revised attributes, i.e., cost, quantity, work schedule, and errors, in order to calculate the level of consistency and complexity factor. The consistency adoptability (ω_1) factor statistically evaluates the extent of consistency after adopting inputs specified in Eq. 1.1.

$$\omega_1 = \sum \frac{IP_{oDyRM} - 0.1}{0.8} . \Delta wt \tag{1.5}$$

Equation 1.5 considers inputs from prior model and generated weight (wt) and considers the lower limit of 0.1 and higher limit of 0.8 in probability theory. The system chooses to consider 0.8 as accomplishing higher consistency factor of 0.9, and 1 is impractical assumption in probability theory. Complexity factor (ω_2) checks the uniformity of the generated values after performing validation to the consistency adoptability factor (ω_1). Hence, the proposed model introduces a mathematical approach using analytical research methodology which ensures

design reusable components to be included in every software project deliveries in order to optimize cost of new production. The first module assists in calculating design reusability, while the second module applies backpropagation algorithm along with incorporation of errors to check the level of consistency in adoptability as well as complexity associated with the proposed model.

The next section discusses the outcome resulting from the proposed model.

1.7 Results and Discussion

This section discusses the outcomes accomplished from the proposed study. The result discussion is carried out in two stages. The first stage of the result discussion is carried out on total amount of reusability factor found from the existing constraints, whereas the second stage of the result discussion is carried out with respect to optimization done in this model. For this purpose, we consider a simple software project of supply chain management with high number of modules developed in Java. The overall numerical outcome is just an approximation of five such types of projects (two projects in supply chain, two projects in customer relationship management, and one project in enterprise relationship management). We also compare the proposed approach with nearly similar existing system to perform benchmarking and assess the effectiveness in the proposed model of computing and optimizing design reusability.

1.7.1 Numerical Outcome of Reusability Estimation Model

The numerical outcome of the proposed system is shown in Tables 1.1, 1.2, 1.3 and 1.4.

Figure 1.5 shows the outcome of the reusability factor ΔD_A considering five design attributes. The parameter X_i and Y_i is used for computing reusability factor ΔD_A. The numerical outcome of the study shows accomplishment of superior value of reusability factor of approximately 73%. In order to estimate the effectiveness of the proposed outcome, we also compare the individual values of the CK metrics with that of Zhou and Leung [30]. Although there are various forms of threshold-based studies formulated by various researchers, the recent study conducted by Antony [31] has shown that the study done by Zhou and Leung [30] has direct interpreted values of threshold and was also found to be referred by almost 169 researchers till date and is one of the latest thresholds formulated till date.

We use statistical analysis to explore the cost of the individual CK metrics. For better comparative analysis, we evaluate the accomplished values of proposed model with respect to the standard threshold values fixed by Zhou and Leung [30]. Our outcome shows that almost all the CK metrics considered for the study

Table 1.1 Reusability calculations

D_A	k (days)	$(k-1)$	Δe	X_i (Eq. 1.1)	Y_i (Eq. 1.2)
5	20	19	8	960	912
5	40	39	8	1920	1872
5	60	59	8	2880	2832
5	120	119	8	5760	5712
				$X = 11{,}520$	$Y = 11{,}328$

Table 1.2 Reusability calculations considering constraint

Jr. Dev (Th = 2)		Total dev duration (months)	Actual development duration (months)	N	D
Two clients	Two projects	6 + 8	4 + 6	364	260
Two projects		14	10		
Sr. dev (Th = 3)		Total dev duration (months)	Actual development duration (months)	N	D
Three clients	Three projects	6 + 8 + 10	4 + 6 + 8	624	468
Three projects		24	18		
$N = N$(Jr. Dev.)+N (Sr. Dev.) = 364 + 624 = 988			$\|D\| = \|D\|$(Jr. Dev.)+$\|D\|$ (Sr. Dev.) = 260 + 468 = 988		

Table 1.3 Reusability calculations considering duration of development

N(days)=	988
$\|D\|$=	728
Z=	260
ΔD_A	0.738461538

accomplished lower value, which is quite good for reusability factor. Lower value of WMC (2.30) will also mean lower number of classes leading to fewer complexes in accessing the program. Similar pattern of RFC (2.44) is also found in the comparative analysis. Smaller values of DIT (2.77) accomplished also show extremely lesser deeper class in hierarchy rendering it less complex. Even the CBO value (2.57) is also less than the maximum limit of 8 thereby reducing the dependencies of the inter-classes. Finally, moderate value of NOC (3.78) has kept a good balance between reuse and less probability of testing the code. It is because if NOC value is high, it ensures reusability, but at the same time, it minimizes reliability and requires more number of testing of classes. Hence, the proposed outcome shows a very good balance between the reusability and other performance of classes.

Table 1.4 Numerical analysis

Set no.	C	Q	WS	E	CAF_{exp}	CAF_{pred}	CF_{exp}	CF_{pred}
2	0.325	2	3	1	0.843	0.843	6.8	6.8
18	0.5	2.5	3	1	0.92	0.921	4.9	4.9
31	0.375	2	3	2	0.853	0.850	6.9	6.8
52	0.45	3	3	2	0.852	0.853	7.4	7.3
72	0.5	2.5	3	3	0.876	0.875	7	7.0
95	0.4	2.5	3	4	0.775	0.775	10.5	10.5
108	0.5	3	3	4	0.817	0.817	8.6	8.7
124	0.45	2.5	2	1	0.89	0.891	5.8	5.7
130	0.375	3	2	1	0.838	0.841	7.2	7.2
137	0.325	2	2	2	0.805	0.808	8.2	8.2
153	0.5	2.5	2	2	0.877	0.876	6	5.8
165	0.35	2	2	3	0.789	0.789	8.9	8.9
172	0.3	2.5	2	3	0.748	0.744	11.6	11.4
197	0.475	2	2	4	0.822	0.823	8.2	8.2
204	0.425	2.5	2	4	0.765	0.763	10.1	10.1
219	0.35	2	1	1	0.854	0.854	6.5	6.4
233	0.475	2.5	1	1	0.905	0.904	5.2	5.3
242	0.475	3	1	1	0.888	0.890	6.1	6.1
248	0.4	2	1	2	0.826	0.824	6.9	6.8
252	0.5	2	1	2	0.882	0.882	5.6	5.6
276	0.425	2	1	3	0.805	0.810	8.1	7.9
297	0.5	3	1	3	0.773	0.772	8.5	8.6
307	0.3	2.5	1	4	0.674	0.673	13.9	13.8
324	0.5	3	1	4	0.734	0.737	10.1	10.1

C cost, *Q* quantity, *WS* work schedule, *E* % errors, CAF_{exp} consistency adoptability factor (experimental), CAF_{pred} consistency adoptability factor (predicted), CF_{exp} complexity factor (experimental), CF_{pred} complexity factor (predicted)

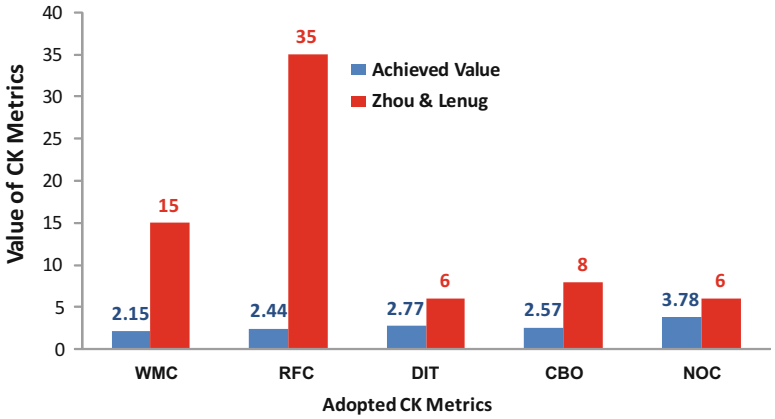


Fig. 1.5 Comparative analysis of proposed and existing system

1.7.2 Numerical Outcome of Optimized Model

The numerical outcome of optimization model is highlighted in Table 1.4. The outcome shows the experimental and actually predicted value of consistency in adoptability and complexity factor. Out of total set of 324 rows of observation, we show only the significant reading in Table 1.4 with different value of costs quantity, work schedule, and error. The calculation of cost, quantity, and work schedule is carried out using statistical analysis of previous model (t-test, analysis of variance, regression test, etc.). The proposed model considers 1–4% of errors in numerical analysis of optimized model. In multilayered perceptron, error computation is carried out by using subtracting desired result with actual result. We use curve fitting toolbox of Matlab to obtain this objective of implementing optimization model. Our experiment uses Levenberg-Marquardt backpropagation algorithm using the toolbox. Usually the error value will lie between 0 and 0.05 using the toolbox. Hence, we consider practical error values, e.g., 0–0.05, as according to probability theory if p value lies between 0 and 0.05, it is considered as ideal value for observation. Hence, we choose the error inclusion to be 0–0.04, which after percentage conversion looks like 1–4%. A closer look into the tabulated data also shows higher accomplishment of software consistency. The overall runtime of the application takes around 2.3 s for a five-project data totaling to 975 MB of file size. Therefore, depending on higher degree of consistency value and lower values of complexities, it can be concluded that proposed model score is well in terms of software reusability with better consistency factor.

1.7.3 Key Process Information

The key process information for the proposed design are basically (i) evaluating mean value for feature attribute CK and (ii) applying backpropagation and Levenberg-Marquardt optimization algorithm. As the proposed model performs reusability of design considering constraint formulations of number of developers and number of clients, cost, and errors, the accuracy depends on assumption of formulation of such constraints. However, even presences of minor errors are rectified in the optimization process using multilayered perceptron as core key process of enhancing the design reusability. Another key process is the adherence to the threshold value of design reusability considering only five CK metrics (i.e., CBO, RFC, WMC, DIT, and NOC). A significant process of the proposed model is basically the consistency measurement with reduced computational complexity. The entire process takes only 1.27–4.76 s to execute for normal operating system without storing any transactional computational data. Hence, the proposed model is highly computationally efficient model.

1.7.4 Threat to Validity

The validity of the proposed model is based on how much is the reduced value of operational cost maintained by its design scheme. It will also mean that as the proposed model adopts optimization, hence, its value of design reusability and consistency has to be higher than any existing technique. We have seen from prior subsection that the proposed model offers better performance of software metric suite with respect to existing study of Zhou and Leung [30]. The entire design and development of the proposed reusability concept are on the basis of the three real-time constraint formulations. The proposed model is less prone to internal threats and moderately prone to external threats. In case new constraint is added apart from three core constraints, the study outcome may slightly differ. However, variance will be minimal as it is compensated by considering external error in the optimization process.

1.8 Conclusion

Developing software product with reusable design brings much to return to IT industries. Reusability has several direct or indirect factors like cost, efforts, and time toward software development. Reusability is the probability that the existing concepts will be used in other scenarios with little modification or no change in their functionality. It is a technology for improving software quality and productivity. Here we optimized three real-time constraints such as work schedule, cost, and effort attribute to measure the external quality attribute reusability. The proposed model offers the solution to finding the extent of design reusability for any open-source software projects for any domain. This phenomenon potentially assists the stakeholder to consider the situation which also assists in risk analysis and cost prediction. As the study uses learning algorithm, so the technique can be said to possess an intelligence to understand the uncertainties and errors in computational values of design reusability. This fact will mean that this technique could be easily adopted by any organization on any domain of work in software project development. Finally, the outcome of the study is found to offer a better alignment between predicted and estimated outcomes during optimization operations. However, the validity of the entire study is high as long as a new variant of real-world constraint is not introduced. We use probability theory to assume a new variant as external error which takes the value between 0 and 1, which is quite easier to scale. Hence external threat to validity of the proposed model is quite moderate. Hence, our future work will be to check for further enhancement of this model. At present, we used nonlinear optimization principle to maintain a well balance between design reusability estimation, consistency, and reduced complexity. Our future study will be therefore in the direction to further escalate the approach with new performance parameters.

References

1. Selvarani R, Nair TRG (2009) Software reusability estimation model using metrics governing design architecture, International book: knowledge engineering for software development cycles: support technologies and applications. Engineering Science Reference, IGI Publishing, New York. doi:10.4018/978-1-60960-509-4.ch011
2. Hooper JW, Chester RO (1991) Software reuse: guidelines and methods. Springer Science & Business Media
3. Selvarani R, Mangayarkarsi P (2015) Reviewing the significance of software metrics for ensuring design reusability in software engineering. Int J Comput Sci Commun Netw 4 (6):208–213
4. Alonso D, Ledesma FS, Sanchez P (2014) Models and frameworks: a synergistic association for developing component-based applications. Hindawi Publishing Corporation
5. Thakral S, Sagar S, Vinay (2014) Reusability in component based software development – a review. World Appl Sci J 31(12):2068–2072
6. Soora SK (2014) A framework for software reuse and research challenges. Int J Adv Res Comput Sci Softw Eng 4(10)
7. Tahir M, Khan F, Babar M, Arif F, Khan S (2016) Framework for better reusability in component based software engineering. J Appl Environ Biol Sci 6(4S):77–81
8. Ahmaro IYY, Yusoff, M Z A, Abualkishik M (2014) The current practices of software reusability approaches in Malaysia. IEEE Malaysian Software Engineering Conference
9. Singh S, Singh R (2012) Reusability framework for cloud computing. Int J Comput Eng Res 2 (6):169
10. Basha N, Md J, Moiz SA (2012) A methodology to manage victim components using CBO measure. Int J Softw Eng Appl 3(2):87
11. Zhu F, Yao Y, Chen H Yao F (2014) Reusable component model development approach for parallel and distributed simulation. Hindawi Publishing Corporation
12. Xiao X, Lina Y, Junwei L (2009) The reuse policy in developing multi-agent system. Int J Distrib Sensor Netw 5:82
13. Gupta C, Rathi M (2013) A meta level data mining approach to predict software reusability. Int J Inf Eng Electron Bus 6:33–39
14. Singh PK, Sangwan OP, Singh AP, Pratap A (2015) A framework for assessing the software reusability using Fuzzy logic approach for aspect oriented software. Int J Inf Technol Comput Sci 02:12–20
15. Monga C, Jatain A, Gaur D (2014) Impact of quality attributes on software reusability and metrics to assess these attributes. IEEE Advance Computing Conference
16. Geertsema B, Jansen S (2010) Increasing software product reusability and variability using active components: a software product line infrastructure. In: ACM-proceedings of the fourth European conference on software architecture
17. Lania A, Quintinoa T, Kimpeb D (2006) Reusable object-oriented solutions for numerical simulation of PDEs in a high performance environment. Sci Program 14:111–139
18. Nuseibeh B, Easterbrook S, Russo A (2000) Leveraging inconsistency in software development. Inst Electr Electron Eng 33(4):24–29
19. Andreou AS Papatheocharous (2015) Towards a CBSE framework for enhancing software reuse: matching component properties using semi-formal specifications and ontologies, In: Evaluation of novel approaches to software engineering
20. Bombonatti D, Goulão M, Moreira A (2016) Synergies and tradeoffs in software reuse – a systematic mapping study. Software: practice and experience
21. Muller G (2003) Software reuse; caught between strategic importance and practical feasibility. Embedded Systems Institute, Article as part of the Gaudí project
22. Rezaee A, Pajohesh M (2016) Creating an environment for reusable software research. J Harmonized Res (JOHR) 4(2):90–92

23. Oliveira J, Fernandes E, Souza M, Figueiredo E (2016) A method based on naming similarity to identify reuse opportunities
24. Sharmaa S, Kumar A, Kavitaa (2016) A design based new reusable software process model for component based development environment. Elsevier, Int Conf Comput Model Secur 85:922–928
25. Stefi A, Lang K, Hess T (2016) A contingency perspective on external component reuse and software project success. A contingency perspective on external software reuse
26. Subedha V, Sridhar S (2012) Optimization of component extraction for reusable software components in context level–a systematic approach. Proc Eng 38:561–571
27. Tibermacine C, Sadou S, That MTTT, Dony C (2016) Software architecture constraint reuse-by-composition. Futur Gener Comput Syst 61:37–53
28. Eclipse Metrics (2016) http://metrics.sourceforge.net
29. Chauvin Y, Rumelhart D (2013) Backpropagation: theory, architectures, and applications. Psychology Press
30. Zhou Y, Leung H (2006) Empirical analysis of object – oriented design metrics for predicting high and low severity faults. IEEE Trans Softw Eng 32(10):771–789
31. Antony PJ (2013) Predicting reliability of software using thresholds of CK metrics. Int J Adv Netw Appl 4(6)

Chapter 2
Estimation of Costs and Time for the Development of Distributed Software

Manal El Bajta, Ali Idri, Joaquín Nicolas Ros, José Luis Fernandez-Aleman, and Ambrosio Toval

2.1 Introduction

Most of today's software development organizations aspire to save time and reduce costs. Therefore, globally distributed environment has invaded the software development industry. The strategy of distributed software development generates many benefits that support the development of software product in an effective way, but this strategy still faces many challenges which may hinder the success of globally distributed software development projects. In this context, a significant number of projects failed to deliver within time and budget in globally distributed environment [1]. Thus, managing the globally distributed environment is a key characteristic. However, in order to successfully plan software development projects' activities, it is important to sustain a high level of accuracy to cost and time estimation methods.

Developing software products in a cost-effective way is the overwhelming objective of many organizations. In addition, the ultimate goal is the accurate estimation of the required amount of effort for the completion of each project. Many research studies indicate that projects without realistic planning and accurate estimation are often beyond their allocated budget and the proposed completion time [2–4].

The drivers involved in the distributed environment are investigated with respect to four aspects: (1) software product, (2) personnel attributes, (3) computer

M. El Bajta (✉) • A. Idri
Software Project Management Research Team, ENSIAS, Mohammed V University, Rabat, Morocco
e-mail: manal.elbajta@gmail.com

J.N. Ros • J.L. Fernandez-Aleman • A. Toval
Software Engineering Research Group, Regional Campus of International Excellence, "Campus Mare Nostrum", University of Murcia, Murcia, Spain

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_2

attributes, and (4) project attributes [5]. We also suggest that distributed software development projects' success is never isolated to one particular driver.

Although there are many methods and techniques available to assist in creating distributed software project effort estimates, they are still far from the required accuracy. Several authors concerned with software development have given varied suggestions for these inaccuracies and ways to overcome some of them [6, 7]. In contrast, this chapter focuses on ways in which existing effort estimation methods can be tailored to account for global software development. It investigates the influence of the different factors that affect the effort estimation method's accuracy in the context of globally distributed software development projects. Furthermore, this chapter presents the effort estimation methods based on the treated factors.

The chapter is structured as follows: Sect. 2.2 presents the globally distributed environments. Section 2.3 reports the software effort estimation process. Section 2.4 outlines software cost/time estimation techniques for global software development (GSD). Section 2.5 discusses the main cost and time drivers. Section 2.6 presents the risk analysis; finally, the conclusions and future work are presented in Sect. 2.7.

2.2 Globally Distributed Environment (GSD)

GSD refers to software development that is done by multiple teams in different geographic locations. The teams are separated physically, and they are located in different countries within one region or around the world. The teams can either be from one organization or from multiple different organizations (outsourcing) [8].

Global software development is usually considered to be much more difficult than collocated software development given the many different challenges related to the software development in a globally distributed setting. These challenges include negative impact of physical distance, cultural differences, and many other complexity factors which are elaborated in the following subsections [9, 10].

Past studies have shown that tasks take about 2.5 times longer in distributed setting than in collocated setting [11, 12]. Other studies reported that about 40% of GSD projects fail to deliver the expected benefits, due to the lack of theoretical basis and difficult complications in GSD project [13, 14]. On the other hand, Teasley et al. [15] reported that in collocated teams, productivity and job satisfaction are much higher than projects that do locate the entire project team in a war room.

The additional activities and difficulties in global software development require additional effort for substantial planning, coordination, and control overhead in the day-to-day governance of global software development. This additional effort should be considered in the time and cost estimation. Hence the time and cost estimation in GSD is more complex than in local development.

2.2.1 Challenges

Although GSD offers several benefits, the distributed work has also many challenges (Table 2.1). If globally distributed software projects are not managed neatly, then they are likely to turn any company into a loss-making business [16]. That means that there are many challenges associated with global software development. Physical separation among project members has diverse effects on many levels. The following factors have been gathered from research literature [17] to have an impact on the amount of effort and cost required for global software development.:

- *Geographic distance*: Software development, particularly in the early stages, requires much communication, coordination, and control [18]. Geographical distance is a measure of the effort required for one actor to visit another and can be seen as reducing the intensity of communication [19], especially when people experience problems with media and have difficulties finding a sufficiently good substitute for face-to-face interaction [20]. Kraut and Streeter [21] found that formal communication is useful for routine coordination, while informal communication is needed to face uncertainty and unanticipated problems, which are typical of software development. They observed that the need for informal communication increases dramatically as the size and complexity of the software increase. In a large software organization, developers can spend on average up to 75 min per day for informal unplanned communication [22]. In general, low geographical distance offers greater opportunity for periods of collocated teamwork.
- *Temporal distance*: Time zone differs among project members when development team is distributed around the world. Temporal distance is a measure of the dislocation in time experienced by two actors wishing to interact [19]. Temporal distance can be caused by time zone difference or time shifting work patterns and can be seen as a factor that reduces opportunities for real-time collaboration, as response time increases when working hours at remote locations do not overlap [23]. Temporal dispersion reduces the possibilities of synchronous interaction, which is a critical communicational attribute for real-time problem solving and design activities. In practice, teams in different time zones have few hours in the work day when multiple sites can participate in a joint synchronous meetings and discussions. Temporal dispersion can also make misunderstandings and errors more likely to occur [24].

  This leads to delay in response to asynchronous communication. For example, an e-mail sent from one site arrives after working hours at the destination; as a consequence, the response cannot be sent until the next day begins, and it will be visible to the sender only when he/she comes to office on the following day.
- *Linguistic distance*: The lack of a common native language creates further barriers to communication [25, 26]. Linguistic distance limits the ability for coherent communication to take place [27]. English has become the popular language of GSD [28]. This affects not only the quality of communication but also the choice of communication media. Language skills can impede

communication in more subtle ways. When participants to a conversation have different levels of proficiency, the group with better language skills occupies a position of strength and can appear to be more powerful and thus suppress important communication through unintended intimidation [28]. Further, lack of proficiency in the chosen language can lead to a preference for asynchronous communication, which can be an impediment if video and teleconferencing are important communication media [29].

- *Cultural distance*: GSD requires close cooperation of individuals with different cultural backgrounds which often creates another barrier for efficient work. Cultures differ on many critical dimensions, such as the need for structure, attitudes toward hierarchy, sense of time, and communication styles. These differences have been recognized as major barriers to communication. Culture also affects interpretation of requirements; domain knowledge used to fill in gaps or place requirements in context varies considerably across national culture [30]. Culture also interferes with collaboration when cultural norms result in conflicting approaches to problem solving.
- *Social challenges*: Another fundamental challenge in global software development is the social issues like fear and trust. Fear and distrust can negatively impact the motivation, the desire to work, the cooperation, and the communication and share of knowledge with remove colleagues. Hence, it has a direct bearing on the success of implementing global software development [31]. It is very difficult for individuals and groups to trust and build relationships with people they feel threaten their jobs. On-site teams in expensive countries are fearful of their job security when off-site teams are added in less expensive locations; this creates mistrust to their off-site colleagues as well as their own management's motives. This can result in clear examples of not wanting to cooperate and share knowledge with remote [26, 31].

Table 2.1 Challenges in global software development

	Challenges
Temporal distance	Reduced opportunities for synchronous communication
	Typically increased coordination costs
	Management of project artefacts may be subject to delays
Geographic distance	Face to face meeting difficulties
	Lack of critical task awareness
	Difficulties to convey vision and strategy
Linguistic distance	Knowledge transfer will not occur smoothly
	Language confusion and misunderstandings
Sociocultural distance	Cultural misunderstandings
	Reduced cooperation arising from misunderstanding
	Different perceptions of authority can undermine morale
	Adaptation of managers to local regulations
	Impact on coordination caused by inconsistent work practices

In some cases, wherein people have successfully worked together for up to year in a collocated situation, once a virtual team strategy was fully implemented, these problems soon came to the fore.

2.2.2 Benefits

This section identifies the main benefits that have been associated with global software development.

2.2.2.1 Cost Savings

One of the most obvious reasons for organizations to embark on a challenging and risky endeavor such as GSD is, not surprisingly, the potential to reduce development costs. By moving parts of the development work to low-wage countries, the same work can be done for a fraction of the cost [32]. The basis for this benefit is that companies are globalizing their software development activities to leverage cheaper employees located in lower-cost economies. This has been made possible by the deployment of cross-continental high-speed communication links enabling the instantaneous transfer of the basic product at hand: software.

The difference in wages across regions can be significant, with a US software engineer's salary being multiple times greater than that of a person with equivalent skills (at least parts) from Asia or South America. However, this seems to be rising, and there has been hyper-growth in local IT employment markets such as in Bangalore. It is our experience that companies are now looking at alternative locations, which offer more acceptable attrition rates with the continued promise of cheaper labor.

2.2.2.2 Reduced Time

Having developers located in different time zones can allow organizations to increase the number of daily working hours in a "follow-the-sun" development model which can decrease cycle time. Time zone effectiveness is the degree to which an organization manages resources in multiple time zones, maximizing productivity by increasing the number of hours during a 24-h day that software is being developed by its teams. When time zone effectiveness is maximized to span 24 h of the day, this is referred to as the "follow-the-sun" development model. This is achieved by handing off work from one team at the end of their day to another team located in another time zone. The approach can aid organizations which are under severe pressure to improve time to market [11].

2.3 Software Effort Estimation Process

In software project management, effort estimation is the process of developing an approximation of the monetary and temporal resources needed to complete project activities [33]. Usually software is developed in projects, and hence software cost and time estimate can be considered as an approximation of the monetary and temporal resources needed to complete software.

2.3.1 Estimation Process

In order to establish an accurate effort estimate for software, a structured approach with significant amount of work is needed. The software effort estimation can be seen as a small size project which needs to be carefully planned, managed, and followed up. Many organizations have different processes for software effort estimation. These processes vary in many aspects, and there does not seem to be one common process which is used in all organizations and in research. The process for software cost and time estimation data gathered from the NASA's *Handbook for Software Cost Estimation* [34] enables us to develop the following table (Table 2.2). It consists on preparing a description of cost analysis requirements, revising its processes and its procedural requirements document and cost/time estimation handbook accordingly.

Most of the software effort estimation models view the estimation process as being a function that is computed from a set of cost drivers. And in most estimation techniques, the primary driver or the most important driver is believed to be the software size. As illustrated in Fig. 2.1, a view of software estimation process, the software requirements are the primary input to the process and also form the basis for the estimation.

2.3.2 Estimation Accuracy

The effort estimation accuracy helps to determine how well or how accurate our estimation is when using a particular model or technique. In addition to the degree of project determination, estimate accuracy is driven by:

- Level of non-familiar technology in the project
- Complexity of the project
- Quality of reference cost estimating data
- Quality of assumptions used in preparing the estimate
- Experience and skill level of the estimator
- Estimating techniques employed
- Time and level of effort budgeted to prepare the estimate
- The accuracy of the composition of the input and output process streams

Table 2.2 Software cost estimation process from NASA

Number	Action	Description
Step 1	Gather and analyze software functional and programmatic	Analyze and refine software requirements, software architecture, and programmatic constraints
Step 2	Define the work elements and procurements	Define software work elements and procurements for specific project
Step 3	Estimation software size	Estimate size of software in logical Source lines of code
Step 4	Estimate software effort	Convert software size to software development effort
Step 5	Schedule the effort	Determine length of time needed to complete the software effort
Step 6	Calculate the cost and time	Estimate the total cost and time of the software project
Step 7	Determine the impact of risks	Identify software project risks, estimate their impact, and revise estimates
Step 8	Validate and reconcile the estimate via models	Develop alternate effort, schedule, and cost estimates to validate original estimates and to improve accuracy
Step 9	Reconcile estimates, budget, and schedule	Review above size, effort, schedule, and cost estimates and compare with project budget and schedule
Step 10	Review and approve the estimates	Review and approve software size effort, schedule, and cost estimates
Step 11	Track, report, and maintain the estimates	Compare estimates with actual data

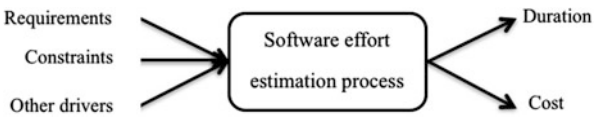


Fig. 2.1 View of software estimation process

We can assess the performance of the software estimation technique by the following two mechanisms:

2.3.2.1 Mean Absolute Error (MAE)

Mean of absolute error (*MAE*) (Eq. 2.1) [35] is computed by averaging the total of absolute errors (*AE*) (Eq. 2.2).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} AE_i \tag{2.1}$$

$$AE_i = |e_i - \widehat{e}_i| \tag{2.2}$$

2.3.2.2 Mean Magnitude of Relative Error (MMRE)

MMRE is defined in Eq. 2.3. This measure is derived from the magnitude of the relative error (*MRE*) as shown in Eq. 2.4. This *MRE* criterion has been criticized by some researchers for being biased toward underestimates, which makes it not significant for being an accuracy measure [36, 37].

$$MMRE = \frac{1}{n}\sum_{i=0}^{n} MRE_i \tag{2.3}$$

$$MRE = \frac{AE_i}{e_i} \tag{2.4}$$

where e_i and $\widehat{e}_i$ are the actual and predicted effort for the ith project.

Each of the error calculation techniques has advantages and disadvantages. For example, absolute error fails to measure the size of the project especially in GSD context, and mean magnitude of relative error will mask any systematic bias (do not know if the estimation is over or under).

2.4 Software Cost/Time Estimation Techniques for GSD

The cost/time estimation has been in the focus of software engineering research for many decades, and hence a high number of different estimation techniques have been developed [38–40]. Unfortunately most of the techniques for software cost estimation have been developed before the recent trend on global software development. Many techniques assume that the software is developed locally, and therefore they do not take into account the additional challenges for the development of distributed software [41, 42].

Estimation for the development of distributed software differs from estimation of local software development at least in two different ways. Firstly, there is a large overhead effort caused by several factors such as language differences; cultural barriers, or time shifts between sites; etc. Secondly, many factors (such as the skills and experience of the workforce) are specific and cannot be considered globally for a project. In many projects, the development sites have very different characteristics, and thus the productivity and cost rate is different between sites.

In the recent research, techniques used to estimate project effort and task duration in distributed context [43] include expert judgment, estimation by analogy, and algorithmic models (i.e., COCOMO II, SLIM, and recently function point analysis-based models) [41].

2.4.1 Expert Judgment

Experts' judgment is one of the methods by which assessors conduct their effort estimation via using their expertise and their logical reasoning to estimate the required amount of effort needed to develop a software product. The accuracy of this method mainly depends on the skills, knowledge, and experience of the assessors to estimate the required amount of effort to complete a given project. Expert judgment can be very accurate, but it fails to provide an objective and quantitative analysis of what are the factors that affect effort and duration in GSD context, and it is hard to separate real experience from the expert's subjective view [44]. The accuracy of the estimates depends on how closely the project correlates with past experience and the ability of the expert to recall all the facets of historic projects.

2.4.2 Estimation by Analogy

Estimating by analogy means comparing the proposed project to previously completed similar project, where the project development information is known. Actual data from the completed projects are extrapolated to estimate the proposed project. This technique is relatively straightforward. Actually in some respects, it is a systematic form of expert judgment since experts often search for analogous situations so as to inform their opinion. The methodology that should be followed to succeed the estimations by analogy involves characterizing the proposed project, selecting the most similar completed projects whose characteristics have been stored in the historical data base, and deriving the estimate for the proposed project from the most similar completed projects by analogy [41, 45].

2.4.3 Algorithmic Models

The algorithmic methods are designed to provide some mathematical equations to perform software estimation. These mathematical equations are based on research and historical data and resort to inputs such as source lines of code, number of functions to perform, and other cost/time drivers such as project effort, design methodology, task allocation, team size, etc. The algorithmic methods have been largely studied and offer several advantages such as generating repeatable estimations, refining and customizing formulas, supporting a family of estimations or a sensitivity analysis, and calibrating previous experience. Models such as COCOMO II (Constructive Cost Model) and SLIM Model are the most frequently algorithmic methods used in a GSD context [43]. In the following, we present:

2.4.3.1 Constructive Cost Model

One of the popular and extensively used algorithmic models for the estimation of cost and schedule of a developing software was given by Shruti Jain [46] and is known as the Constructive Cost Model (COCOMO) [47, 48]. The parameters and equations that are used in this model are obtained through previous software projects. The size of code is usually given in KLOC (thousand lines of code), and the obtained effort is in person months (PM). The PM represents the number of hours that a person spend to complete a given task presented in a calendar month. COCOMO II deals with variety of factors that influence development of distributed software projects' effort estimation. There are three submodels for COCOMO II: Application Composition Model, Post-architecture Model, and Early Design Model. COCOMO II includes factors in order to steer the effort estimation team to make better approximation based on the influencing factors. These factors are related to organizational and team characteristics. Each factor has values from range of very low to extra high rating level. The weight of scaling factors could divert according to organizations and projects. The following are the equations which COCOMO II proposed to estimate the required effort:

$$PM = A \times \text{Size}^{E} \times \prod_{i=0}^{n} EM^{i} \tag{2.5}$$

where:

- n represents the number of drivers in a GSD context.
- $A = 2.94$ (for COCOMO II). size is estimated by kilo source lines of code (KSLOC) measure.
- $E = B + 0.01 \times \sum_{i=1}^{4} \text{Factor}$
- EM represents the effort multiplier; $B = 0.91$ for COCOMO II.

$$\text{Duration} = C \times PM^{D+0.2\times(E-B)} \tag{2.6}$$

where $C = 3.67$, $D = 0.28$, and $B = 0.91$

2.4.3.2 Slim

SLIM [49] is an algorithmic method that is used to estimate effort and schedule for projects. The underlying reason for developing SLIM is to measure the overall size of a project based on its estimated SLOC. It is represented by two equations: Eq. 2.7 for allocating productivity parameter (PP), expressed in man years, which would be required in Eq. 2.8 for calculating effort.

$$PP = \frac{\text{Size}_{SLOC}}{\left(E_{\text{Man, Year}}/B^{1.13}\right) \times \text{Duration}\left(Y^{4/3}\right)} \tag{2.7}$$

$$E_{\text{Man, Year}} = \left[\frac{\text{Size}_{SLOC}}{PP \times \left(\text{Duration}_{\text{Years}}\right)^{3/4}}\right]^3 \tag{2.8}$$

where:

- $E_{\text{Man,Year}}$ represents the amount of effort required to accomplish a given task in a man-year unit.
- Y is the development time in years.
- B is a special skill factor and is based on size and duration.

Muhairat et al. [43] investigated the effects of different factors on the accuracy of effort estimation methods in GSD environments. Precisely, COCOMO II and SLIM methods of estimating project efforts were considered. They discovered that the estimation methods were less accurate in determining the actual time of completion of some software development projects. The main factor that affected this outcome included the project environment. They concluded that developing software in a GSD environment always requires more effort and time to complete.

2.5 Cost and Time Drivers

As already known, the distributed software development introduces new challenges in the software engineering area. In order to have a better project planning for multisite projects, it is important to identify the main drivers that can increase the project's effort. This section aims to describe these main effort drivers and their impacts on a distributed project.

Analyzing the main researches found in the literature and the feedback from project managers about the impact on project duration and effort would enable us to suggest some effort drivers for distributed software development projects. We present the effort drivers extracted from theoretical research and interview analyses. The effort drivers are split into four categories depicted in Table 2.3 [41]: product, platform, personnel, and project factors. Therefore, the effort drivers tend to be measures of system size and complexity, personnel capabilities and experience, hardware constraints, and availability of software development tools.

2.5.1 Product Factors

The product factors are determined by the novelty of the software to be developed. This category factors indicates the degree of innovation which is directly

Table 2.3 Software drivers

Category	Drivers
Product	Code size
	Reuse
	Product complexity
Platform	Design and technology newness
	Time zone
	Platform volatility
Personal	Team size
	Team culture
	Team trust communication
	Development productivity
Project	Project effort
	Project management effort
	Process model
	Task allocation
	Work pressure
	Client involvement
	Work dispersion

proportional to the level of spontaneous communication, the need for specific domain knowledge, and the frequency of unforeseen changes. Another important factor is the work assignments that have to be carefully crafted and taking into account the organizational structure and the functional coupling among software units [50]. Therefore, the architecture has major influence on the efforts needed to coordinate the development phase. Indicators for the degree of architectural adequacy might be modularity, interface match and dependencies, and communicability of the architecture. Examples of product cost drivers of COCOMO II are:

- *Required Software Reliability (RELY)*: This is the measure of the extent to which the software must perform its intended function over a period of time.
- *Date Base Size (DATA)*: Measure to capture the effect of large data requirements have on product development.
- *Required Reusability (RUSE)*: This cost driver accounts for the additional effort to construct components intended for reuse on the current or future projects.
- *Documentation Match to Life Cycle Needs (DOCU)*: Measure of the suitability of the project's documentation to its life cycle needs.

2.5.2 Platform Factors

The platform factor refers to the target-machine complex of hardware and infrastructure software. Platform products have more demanding task characteristics than derivative products. Specifically, platform projects undertake development of

greater levels of new technology and have higher levels of project complexity. Examples of platform cost drivers of COCOMO II are:

- *Execution Time Constraint (TIME)*: This is a measure of the execution time constraint imposed upon a software system.
- *Main Storage Constraint (STOR)*: This is a rating that represents the degree of main storage constraint imposed on a software system or subsystem.
- *Platform Volatility (PVOL)*: This is a measure of the complex of hardware and software.

2.5.3 Personnel Factors

As for the personnel factors, it includes cultural fit mainly related to closeness of team members' mental models [51] which is influenced by the combination of countries involved, the international experience of the teams, etc., skill level measured by educational level and language skills indicating the formal abilities of remote team(s) [50], shared understanding embodied by tacit knowledge that is required indicating the level of completeness of documentation and specification and the common knowledge about goals, and finally information sharing constraints representing competitive restrictions on information distribution, e.g., when working with external subcontractors or in security-sensitive environments. Examples of personnel cost drivers of COCOMO II are:

- *Programmer Capability (PCAP)*: Current trends continue to emphasize the importance of highly capable analysts.
- *Applications Experience (AEXP)*: This rating is dependent on the level of application experience of the project team developing the software system.
- *Language and Tool Experience (LTEX)*: This is a measure of the level of programming language and software tool experience of the project team developing the software system.

2.5.4 Project Factors

Regarding project factors, we might consider the novelty of collaboration model by analyzing the initial cost for the search of offshore partners and contract negotiation. The tools and infrastructure represent the homogeneity of the tool chains used in all sites and potential ramp-up costs for setting up the infrastructure in remote sites and finally the physical distance representing the potential overlaps of working time and, accordingly, the intensity of use of asynchronous communication media and collaboration tools [52]. Examples of project cost drivers of COCOMO II are:

- *Multisite Development (SITE)*: The assessment and averaging of two factors, site collocation and communication support.
- *Required Development Schedule (SCED)*: This rating measures the schedule constraint imposed on the project team developing the software.

2.6 Risk Analysis

In order to analyze the impact of risk involved in the development of software, the project manager has to identify the risk drivers. Software risk components can be classified as "cost" and "time" risks. The degree of uncertainty that the project budget will be maintained is the cost risk. The degree of uncertainty that the project schedule will be maintained and that the product will be delivered in time is the time risk.

Software risks are managerial issues which should be handled through proper management of the project especially when estimating costs and times. Only expert manger associated with software project office can handle these issues, while a less experienced software manager may lead to un-controlling the risks and ultimately result in the failure of the project. Software risks should be monitored and controlled since the starting phases of the project management life cycle [53].

The GSD is becoming very difficult, complex, and challenging in the context of software project management as the user problem is getting more and more challengeable [19, 54]. In this respect, the risk management in distributed software development is also much complex than in local software development. It particularly has specific concerns that may not be obvious until their impact has been realized. Many projects got failed they did not realize, soon enough, the importance of certain common factors in GSD projects [41]. Table 2.4 presents the potential risks in a GSD project and provides their cost and time impact in this respect.

To systematically identify risks and evaluate appropriate risk mitigation for estimating cost and time in the GSD context, we analyze the features of GSD and then elaborate how they are impacted by risks [55].

Efficiency

Software and IT companies need to deliver promptly and reliably while the competition is literally a mouse click away. Hardly any other business has so low entry barriers as IT and therefore stimulates an endless fight for efficiency along the dimensions of improved cost, quality, and time to profit. GSD clearly helps in improving efficiency due to labor cost differences across the world, better quality with many well-trained and process-minded engineers especially in Asia, and shorter time to profit with following the sun and developing and maintaining software in two to three shifts in different time zones. Risks directly related to the efficiency target are project delivery failures, requirement, and design quality (Table 2.4).

Table 2.4 Risk items and their impacts

	Risk item	Cost/time impact
Efficiency	Late delivery of software	Computer time costs
	Feasibility of requirements	Staffing to conduct analysis
	Feasibility of design	Recruiting and training costs and times
		Added time and cost for review preparation
Flexibility	Lack of management visibility	Added time and cost to prepare inputs for reports
	Lack of a test discipline	Cost and time of using test group

Flexibility
Software organizations are driven by fast changing demands on skills and sheer numbers of engineers. With the development of a new and innovative product, many people are needed with broad experiences. However, when arriving in maintenance, these skill needs look different and manpower distributions are also changing. Such flexible demand cannot anymore be handled inside the enterprise. GSD is the answer to provide skilled engineers just in time and thus allows building flexible ecosystems combining suppliers, customers with engineering and service providers. Directly related risks to the flexibility goal are poor management visibility and distance and culture clashes (refer to Table 2.4).

2.7 Conclusion

There is a strong surge for global software development to countries with lower labor cost. This chapter promotes analysis of project drivers to gain insights into comparing development costs and time for distributed software development projects as compared to collocated projects.

Even though most of the evaluated software effort estimation techniques do not have any of the GSD-related cost/time factors included by default, these techniques are still suitable and applicable for estimation of GSD project with some setup and calibration work. Estimation methods such as estimation by analogy and algorithmic models can be applied to the development of distributed software if the person doing the estimation model setup is experienced in outsourcing. Then, the person would be able to include all necessary cost/time factors into the estimation model.

Also, all expertise-based techniques can be directly applied for GSD projects, but they require experts with experience and knowledge on GSD. The available development of distributed software specific techniques can naturally be also directly applied for GSD projects.

Future work of this research includes on one hand the verification and improvement of the factors of a distributed development project and on the other hand the application of methods on projects while collecting effort data to calibrate the relevance of each project driver.

References

1. Kile JF (2005) The Importance of effective requirements management in offshore software development projects. Doctoral dissertation, Pace University
2. Nguyen V, Steece B, Boehm B (2008) A constrained regression technique for COCOMO calibration. In: Proceedings of the second ACM-IEEE international symposium on empirical software engineering and measurement. ACM, pp 213–222
3. Anderson SD, Molenaar KR, Schexnayder CJ (2007) Guidance for cost estimation and management for highway projects during planning, programming, and preconstruction, vol 574. Transportation Research Board, Washington, DC
4. Wallace L, Keil M (2004) Software project risks and their effect on outcomes. Commun ACM 47(4):68–73
5. Ashiegbu BC, Ahaiwe J (2011) Software cost drivers and cost estimation in Nigeria. Interdiscip J Contemp Res Bus 3(8):431
6. Herbsleb JD (2007) Global software engineering: the future of socio-technical coordination. In 2007 future of software engineering. IEEE Computer Society, pp 188–198
7. Nicholson B, Sahay S (2001) Some political and cultural issues in the globalisation of software development: case experience from Britain and India. Inf Organ 11(1):25–43
8. Gopal A, Gosain S (2010) Research note-the role of organizational controls and boundary spanning in software development outsourcing: implications for project performance. Inf Syst Res 21(4):960–982
9. Holmstrom H, Conchúir EÓ, Agerfalk J, Fitzgerald B (2006) Global software development. challenges: a case study on temporal, geographical and socio-cultural distance. In: 2006 I.E. international conference on global software engineering (ICGSE'06). IEEE, pp 3–11
10. Lanubile F, Damian D, Oppenheimer HL (2003) Global software development: technical, organizational, and social challenges. ACM SIGSOFT Softw Eng Notes 28(6):2–2
11. Herbsleb JD, Moitra D (2001) Global software development. IEEE Softw 18(2):16–20
12. Nguyen T, Wolf T, Damian D (2008) Global software development and delay: does distance still matter? In: 2008 I.E. international conference on global software engineering. IEEE, pp 45–54
13. Betz S, Mäkiö J (2008) Amplification of the COCOMO II regarding offshore software projects. Offshoring of software development: methods and tools for risk management; [OUTSHORE; Proceedings], 33
14. Peixoto CEL, Audy JLN, Prikladnicki R (2010) Effort estimation in global software development projects: preliminary results from a survey. In: 2010 5th IEEE international conference on global software engineering. IEEE, pp 123–127
15. Teasley SD, Covi LA, Krishnan MS, Olson JS (2002) Rapid software development through team collocation. IEEE Trans Softw Eng 28(7):671–683
16. Yadav MS, Prabhu JC, Chandy RK (2007) Managing the future: CEO attention and innovation outcomes. J Mark 71(4):84–101
17. Noll J, Beecham S, Richardson I (2010) Global software development. And collaboration: barriers and solutions. ACM Inroads 1(3):66–78
18. Yadav V (2016) A flexible management approach for globally distributed software projects. Glob J Flex Syst Manag 17(1):29–40
19. Agerfalk PJ, Fitzgerald B, Holmstrom Olsson H, Lings B, Lundell B, Ó Conchúir E (2005) A framework for considering opportunities and threats in distributed software development
20. Smith PG, Blanck EL (2002) From experience: leading dispersed teams. J Prod Innov Manag 19(4):294–304
21. Kraut RE, Streeter LA (1995) Coordination in software development. Commun ACM 38 (3):69–82
22. Perry DE, Staudenmayer NA, Votta LG (1994) People, organizations, and process improvement. IEEE Softw 11(4):36–45

23. Sarker S, Sahay S (2004) Implications of space and time for distributed work: an interpretive study of US–Norwegian systems development teams. Eur J Inf Syst 13(1):3–20
24. Espinosa JA, Nan N, Carmel E (2007) Do gradations of time zone separation make a difference in performance? A first laboratory study. In: ICGSE, pp 12–22
25. Herbsleb JD, Grinter RE (1999) Splitting the organization and integrating the code: Conway's law revisited. In: Proceedings of the 21st international conference on Software engineering. ACM, pp 85–95
26. Niinimaki T, Piri A, Lassenius C (2009) Factors affecting audio and text-based communication media choice in global software development projects. In: 2009 fourth IEEE international conference on global software engineering. IEEE, pp 153–162
27. Casey V, Richardson I (2006) Uncovering the reality within virtual software teams. In: Proceedings of the 2006 international workshop on Global software development for the practitioner. ACM, pp 66–72
28. Lutz B (2009) Linguistic challenges in global software development: lessons learned in an international SW development division. In: 2009 fourth IEEE international conference on global software engineering. IEEE, pp 249–253
29. Lings B, Lundell B, Agerfalk J, Fitzgerald B (2007) A reference model for successful distributed development of software systems. In: International conference on global software engineering (ICGSE 2007). IEEE, pp 130–139
30. Herbsleb JD, Paulish DJ, Bass M (2005) Global software development at siemens: experience from nine projects. In: Proceedings of the 27th international conference on software engineering. ICSE 2005. IEEE, pp 524–533
31. Casey V, Richardson I (2008) The impact of fear on the operation of virtual teams. In: 2008 I. E. international conference on global software engineering. IEEE, pp 163–172
32. Carmel E, Agarwal R (2001) Tactical approaches for alleviating distance in global software development. IEEE Softw 18(2):22–29
33. PMBoK, A (2000) Guide to the project management body of knowledge. Project Management Institute, Pennsylvania
34. Lum K, Bramble M, Hihn J, Hackney J, Khorrami M, Monson E (2003) Handbook for software cost estimation. NASA Jet Propuls Lab JPL D-26303
35. Shepperd M, MacDonell S (2012) Evaluating prediction systems in software project estimation. Inf Softw Technol 54(8):820–827
36. Myrtveit I, Stensrud E, Shepperd M (2005) Reliability and validity in comparative studies of software prediction models. IEEE Trans Softw Eng 31(5):380–391
37. Miyazaki Y, Takanou A, Nozaki H, Nakagawa N, Okada K (1991) Method to estimate parameter values in software prediction models. Inf Softw Technol 33(3):239–243
38. Idri A, azzahra Amazal F, Abran A (2015) Analogy-based software development effort estimation: a systematic mapping and review. Inf Softw Technol 58:206–230
39. Jorgensen M, Shepperd M (2007) A systematic review of software development cost estimation studies. IEEE Trans Softw Eng 33(1):33–53
40. Wen J, Li S, Lin Z, Hu Y, Huang C (2012) Systematic literature review of machine learning based software development effort estimation models. Inf Softw Technol 54(1):41–59
41. El Bajta M, Idri A, Fernández-Alemán JL, Ros JN, Toval A (2015) Software cost estimation for global software development a systematic map and review study. In: Evaluation of Novel Approaches to Software Engineering (ENASE), 2015 international conference on. IEEE, pp 197–206
42. El Bajta M (2015) Analogy-based software development effort estimation in global software development. In: 2015 I.E. 10th international conference on global software engineering workshops. IEEE, pp 51–54
43. Muhairat M, Aldaajeh S, Al-Qutaish RE (2010) The impact of global software development factors on effort estimation methods. Eur J Sci Res 46(2):221–232
44. Jorgensen M (1995) Experience with the accuracy of software maintenance task effort prediction models. IEEE Trans Softw Eng 21(8):674–681

45. Amazal FA, Idri A, Abran A (2014) Software development effort estimation using classical and fuzzy analogy: a cross-validation comparative study. Int J Comput Intell Appl 13 (3):1450013
46. Jain MS (2012) Survey of various cost estimation techniques. Int J Adv Res Comput Eng Technol (IJARCET) 1(7):229
47. Boehm BW, Madachy R, Steece B (2000) Software cost estimation with Cocomo II with Cdrom. Prentice Hall PTR, Upper Saddle River
48. Boehm B, Clark B, Horowitz E, Westland C, Madachy R, Selby R (1995) Cost models for future software life cycle processes: COCOMO 2.0. Ann Softw Eng 1(1):57–94
49. Kemerer CF (1987) An empirical validation of software cost estimation models. Commun ACM 30(5):416–429
50. Sosa ME, Eppinger SD, Rowles CM (2004) The misalignment of product architecture and organizational structure in complex product development. Manag Sci 50(12):1674–1689
51. O'Hara M, Johansen R (1994) Global work: bridging distance, culture and time. Jossey-Bass, San Francisco
52. Sosa ME, Eppinger SD, Pich M, McKendrick DG, Stout SK (2002) Factors that influence technical communication in distributed product development: an empirical study in the telecommunications industry. IEEE Trans Eng Manag 49(1):45–58
53. Boehm BW (1988) A spiral model of software development and enhancement. Computer 21 (5):61–72
54. Tufekci O, Cetin S, Arifoglu A (2010). Proposing a federated approach to global software development. In Digital Society, 2010. ICDS'10. Fourth International Conference on. IEEE, pp 150–157
55. Ebben JJ, Johnson AC (2005) Efficiency, flexibility, or both? Evidence linking strategy to performance in small firms. Strateg Manag J 26(13):1249–1259

Chapter 3
Using COSMIC for the Functional Size Measurement of Distributed Applications in Cloud Environments

Filomena Ferrucci, Carmine Gravino, and Pasquale Salza

3.1 Introduction

Software sizing is a crucial management activity since it supports several other software project management tasks, such as effort/cost estimation, project planning, productivity benchmarking, and quality control. It is widely recognised that the competitiveness of software companies greatly depends on the ability of their project managers to carry out a reliable and accurate software size estimation. To this aim, functional size measurement (FSM) methods have been extensively investigated in software engineering research and are also widely applied in industry. The success of those approaches is mainly due to the fact that sizing is based on the functionality provided to the users, i.e., the Functional User Requirements (FURs), rather than on other software artefacts (e.g., code) that are not available in the early phases of software life cycle when size estimation is especially important [1].

The Function Point Analysis (FPA) was the first FSM method to be introduced in 1979 [2]. Since then, several variants have been proposed (known as first-generation FSM methods) to improve the size measurement or extend its application domain. COSMIC [3] is a second-generation FSM method, being the first to comply to the standard ISO/IEC14143/1 [1]. It is based on fundamental principles of software engineering and measurement theory.

The COSMIC size is essentially obtained by counting the data movements Entry, Exit, Read, and Write [3]. Although the method was conceived for business, real-time, and infrastructure software (or hybrids of these), COSMIC turns out to be applicable to other kinds of software systems (except for those characterised by complex algorithms) provided that suitable guidelines are provided to ensure that

F. Ferrucci (✉) • C. Gravino • P. Salza
University of Salerno, Fisciano, Italy
e-mail: fferrucci@unisa.it; gravino@unisa.it; psalza@unisa.it

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_3

the specific characteristics of those kinds of software are appropriately captured by the measurer. As a matter of fact, recent empirical studies have been conducted in order to verify the capability of COSMIC size to predict the effort needed to develop web applications. Di Martino et al. [4] reported a better predictive capability of COSMIC against first FSM methods, namely, FPA. Moreover, recent studies have investigated the applicability of COSMIC to mobile applications [5, 6]. This domain is rapidly growing, and new software engineering processes, including functional size measurement and estimation methods [7], are required to improve the quality of these applications. Similarly, the explosive growth of cloud computing requires suitable software size measurement approaches able to support project managers in planning, management and control of software suitable for distributed environments.

In this chapter, we analyse various aspects of the use of the COSMIC method to measure distributed applications in cloud environments. The analysis considers the three distinct provision models of the cloud computing stack: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Furthermore, it deals with specific concepts of cloud computing such as orchestrations and load balancing of several components that act together to realise the required functionality and to ensure critical non-functional requirements (e.g., scalability, reliability).

The rest of the paper is organised as follows. We first give an overview of the FSM and COSMIC methods in Section 3.2. In Sect. 3.3, we analyse the different approaches and guidelines present in the literature about functional size measurement of distributed applications in cloud environments. Section 3.4 contains some final remarks and future work.

3.2 Software Size and COSMIC Measurement

COSMIC is a second-generation function size measurement (FSM) method, and it has some main characteristics in contrast to traditional function point methods. In this section, we give an overview of the history of COSMIC with its main features and usage.

3.2.1 FSM Methods

The measures proposed in the literature to size software can be grouped into two main families: the functional and dimensional ones. A functional size measure is defined as ‘a size of software derived by quantifying the Functional User Requirements (FURs)’ [1]. Thus, measures obtained by applying FSMs are particularly suitable for the early phases of the software development process, when only FURs are available. They can be then exploited for tasks such as estimating a project

development effort. Moreover, they can be employed for performing comparisons among projects developed with different platforms, solutions and so on, since they are independent of the adopted technologies. Differently, dimensional size measures basically allow counting structural properties of a software artefact, such as LOCs, the number of web pages and so on. As a consequence, they are strongly dependent on the adopted technological solutions, they can be employed only after the artefact has been developed and often a standard counting procedure is missing [8, 9].

Function Point Analysis (FPA) is considered the first FSM method proposed in the literature. It was introduced by Albrecht in 1979 [10] to have a measure (the function points) able to overcome the limitations of LOCs, by sizing the 'functionality' provided by a software, and from the point of view of end users. FPA is considered a structured method to perform a functional decomposition of the system whose size is the (weighted) sum of unitary elements (its FURs). The idea is that unitary elements can be measured more easily than the whole system. FPA has evolved in many ways. The original formulation was successively extended by Albrecht and Gaffney [10]. Since 1986 FPA is managed by the International Function Point Users Group (IFPUG) and it is named IFPUG FPA (IFPUG, for short), which has been standardised by ISO as ISO/IEC 20926:2009. Nevertheless, since FPA was originally designed from the experience gained by Albrecht on the development of management information systems, some researchers have analysed the applicability of FPA to other software domains [11, 12]. As a consequence of this debate on the application of FPA, many variants of the method have been defined for specific domains, such as MkII function point for data-rich business applications, or full function point (FFP) method for embedded and control systems [13]. All these variants are known and indicated as first-generation FSM methods since they are all based on the original formulation by Albrecht.

In the middle of the 1990s, important issues in the foundations of FPA against the measurement theory were highlighted in different researches. In particular, an improper use of different types of scales was highlighted in many steps of the FPA process. Moreover, the debate has interested how the 'weights' were defined and used in the method [14, 15].

At the end of the 1990s, a group of experienced software measurers formed the Common Software Measurement International Consortium (COSMIC) with the aim of addressing and overcoming the issues highlighted about the application of FPA and for defining a broader measurement framework able to tackle new IT challenges. The result of this investigation was the COSMIC-FFP method, which is considered the first 'second-generation FSM method'. To highlight this concept, the first version of the method was the 2.0. Successively, in 2007 the version 3.0 was characterised by many refinements and standardised as ISO/IEC 19761:2011. The method was named simply COSMIC. The current version of COSMIC is 4.0.1 [3], introduced in April 2015.

3.2.2 The COSMIC Method

The idea underlying the COSMIC method is that, for many types of software, most of the development efforts is devoted to handling data movements from/to persistent storage and users. Thus, a meaningful sight of the system size can be obtained by considering the number of these data movements [3]. As a functional size measurement method, the functional size is defined as the size of the software derived by quantifying the Functional User Requirements (FURs) [3]. FURs describe what the software is expected to do for its users. The standardised measure defined by COSMIC allows quantifying the functional size of the software in terms of COSMIC function point (CFP) units.

One of the main concepts underlying COSMIC is the functional process, which is defined as a set of data movements representing an elementary part of the FURs. A functional user is defined as a (type of) user that is a sender and/or an intended recipient of data in the FURs. Thus, a human or, for instance, an external device as well can play the role of a functional user. Another important concept that allows determining data movements is the boundary, which can be seen as a conceptual interface between the software being measured and its functional users. With these definitions of functional users and boundary, four different data movement types can be considered: an Entry (E) moves data from a functional user to a functional process; an Exit (X) moves data from a functional process to a functional user; a Write (W) moves data from a functional process to persistent storage; a Read (R) moves data from persistent storage to a functional process. One CFP unit is counted per each data movement, and the size of a software can be obtained by summing all the identified data movements.

The measurement process of COSMIC [3] is defined in terms of three phases: the 'Measurement Strategy Phase', the 'Mapping Phase' and the 'Measurement Phase' as depicted in Fig. 3.1.

3.2.2.1 Measurement Strategy Phase

The concept of Measurement Strategy Phase has been introduced in the last current version 4.0 of COSMIC [3], and it is meant to set the key parameters of the measurement: the purpose of measurement, the scope, the functional users and the level of granularity. The purpose defines what the measurement result will be used for; the scope specifies which pieces of software (in terms of FURs) have to be measured; the level of granularity describes how much detailed the documentation about the software is (e.g., in terms of the requirements description or also the structure description). The complete list of parameters can be found in the COSMIC Context Software Model, and it is necessary to carefully define them.

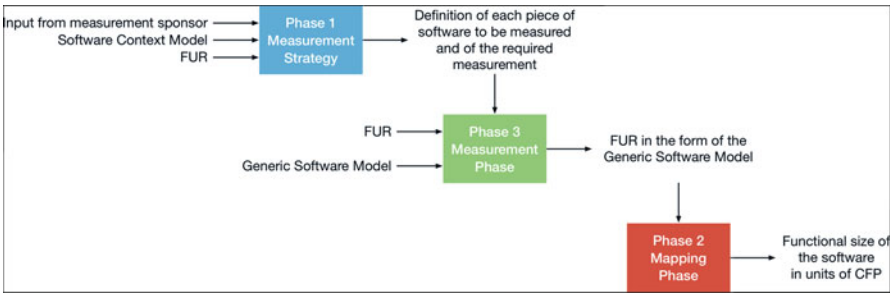


Fig. 3.1 The COSMIC method measurement process

3.2.2.2 Mapping Phase

This phase allows measurers to extrapolate the functional processes from the available FURs of the software being measured. In particular, a technical work has to be performed during Mapping Phase, carefully following the principles and, above all, the rules of the COSMIC method reported in the COSMIC Generic Software Model [3]. The potential functional processes inside the FURs can be identified by measurers looking at each functional process that is started by a triggering Entry and comprises at least two data movements: an Entry plus either an Exit or a Write. Indeed, the Entry of the functional user that starts the functional process can be identified as a triggering Entry. Other three crucial concepts are subprocess, data group, and data attribute. A subprocess may be either a data movement or a data manipulation. The COSMIC manual clearly suggests that the data manipulations inside a functional process are not counted as CFP. They are considered associated with the corresponding data movements. A data group is a distinct, non-empty and non-ordered set of data attributes, where each attribute describes a complementary aspect of the same object of interest. A data attribute is the smallest piece of information, within an identified data group, carrying a meaning from the perspective of the interested FUR. The object of interest is defined as any 'thing' that is identified from the point of view of the FURs. Thus, an object of interest may be any physical thing, as well as any conceptual object or part of a conceptual object in the world of the functional user about which the software is required to process and/or store data.

Four types of data movements are defined: each Entry, Exit, Read or Write is a movement of the data group of a single object of interest. An Entry moves a data group from a functional user across the boundary into the functional process where it is required; an Exit moves a data group from a functional process across the boundary to the functional user that requires it; a Read moves a data group from persistent storage within each of the functional processes that require it; and a Write moves a data group lying inside a functional process to persistent storage. There are only two exceptions: the triggering Entry which can start a functional process without data movement; e.g. in specific enquiry for a list of items, the error/confirmation message that is defined as an Exit for the attention of a human user

that either confirms only that entered data is accepted or only that there is an error in the entered data.

3.2.2.3 Measurement Phase

This Phase defines how to count data movements and it consists in associating a CFP to each data movement. The functional value of the measurement is obtained by considering the amount of all data movements. It is worth noting that the Measurement Phase may become more complex in cases (differently from our work) of aggregating measurement sizes (software stratified into different layers) or when measuring the size of software changes [3].

It is important to mention that the COSMIC community has also proposed approaches for counting the size of software in terms of COSMIC by exploiting approximate counts. We can highlight a couple of situations when there is the need of measuring a functional size approximately [3]: it can happen either early in the life of a project before the FURs have been specified in detail ('early sizing') or when a measurement is needed, but there is insufficient time or resources to apply the standard detailed method ('rapid sizing'). These motivations are not mutually exclusive and contribute to reaching a trade-off between a correct measurement and time and budget available.

3.3 Measuring Distributed Applications in Cloud Environments with COSMIC

As for any kind of software, a project manager that intends to provide a functional size of a project involving cloud environments must first define from which perspective the distributed application needs to be considered. There are three provision models of cloud computing, as shown in Fig. 3.2, in which the customers and the cloud vendors play a different role and have different responsibilities in the management of various aspects: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

In IaaS, the cloud vendor owns the hardware and network, and it is responsible for housing, running, and maintenance aspects. The customers can use virtual infrastructures (i.e., cloud instances), which run on physical resources but that can be created, reconfigured, resized and removed in a few moments based on the customers' or distributed applications' needs. The cloud instances consist of virtual machines for which the customer has full control of the environment. Before deploying a distributed application, the customer needs to install an operating system (OS) and the software stack. The main targets of IaaS are the system admins.

In PaaS, a platform is provided to customers that can run their applications and business in a distributed environment, without having to deal with lower-level

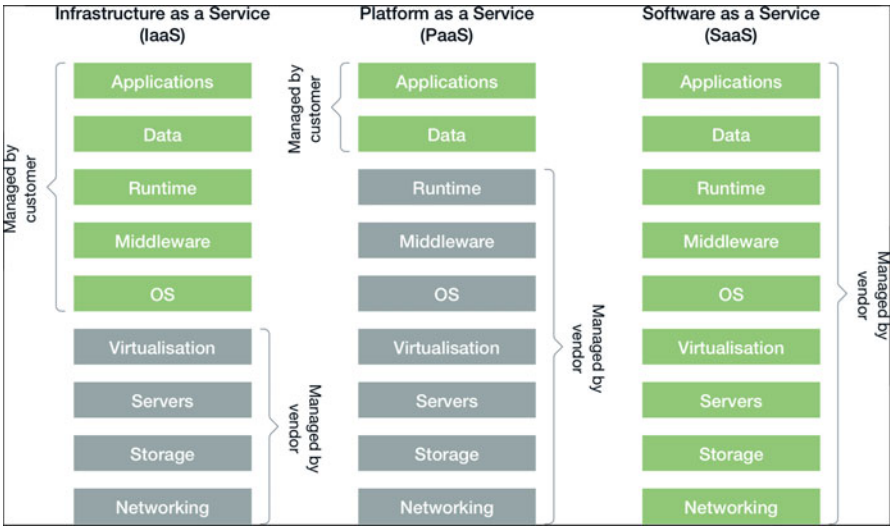


Fig. 3.2 Cloud provision models

requirements such as configuration, security, and management aspects. PaaS abstracts away all the aspects related to hardware decisions. Some examples of PaaS are database, development tools, and web server services. The target of PaaS is the developer, who writes codes that can run on the provided platforms.

SaaS is the top layer of cloud computing. The cloud vendor supplies software to customers in the form of a service such as e-mail clients, virtual desktop and communication services. The target user of SaaS is the end user, and it generally works on a subscription model, which means that the customer pays for the duration of time she/he uses the services. From an architectural point of view, distributed applications at PaaS and SaaS levels can be considered as service-oriented architecture (SOA) services, and therefore every existent contribute for SOA is also valid for distributed applications in the cloud environments.

The literature offers different approaches and guidelines [16–18] related to two main possible interactions: between cloud vendor components and customers and between SOA services. In the following, we collect and describe these literature contributes [16–18], giving specific considerations for cloud environments.

3.3.1 Measuring the Interaction Between Cloud Customers and Cloud Vendor Components

Schmietendorf et al. adapted the COSMIC method for cloud system size measurement [16]. In particular, they focused on the interaction between a cloud customer and the system provided by a cloud vendor.

For the authors, the COSMIC measurement strategy is defined considering the size of a chosen part of the developed cloud system for the purpose of measurement. They provided a rich list of possible size aspects such as the size of the involved services, the interaction between them, the customers, the resources, etc.

The scope of measurement is identified as the application of a service in the cloud system based on chosen characteristics (e.g., scalability, low cost), whereas the decomposition and granularity are established at five different levels, described below. The data movement-based interactions in the different cloud system levels correspond to the functional processes.

The FURs in the scope of measurement are based on the following characteristics:

- The on-demand services that are instantiated on the cloud system are controlled by triggering events and special object of interest like costs and resources.
- The network is the channel with which services communicate as separate layers, measured by means of COSMIC Entry and Exit data movements.
- Instead, Reads and Writes measure data movements from/to persistent storage with which the resource pooling is performed.
- The elasticity factor involves functional processes of service scaling on data groups like ‘storage size’ and ‘location’ requirements.
- The measured service itself supports the functional processes based on the object of interest like billing and service level agreement (SLAs) terms.

Thus, they investigated the cloud systems defining the following five levels of functionality (see also Fig. 3.3) and their interactions [16]:

1. The base level of virtualisation, representing the different virtual instances producing a service
2. The level of instances clustering on machine level
3. The level of multi-data centres, representing the aspects of multiple redundancies of machines on differently located data centres
4. The service level of the SOA, where the distributed application is run and the measurement is performed in terms of service interactions
5. The service chain levels if the application is distributed through different cloud vendors

Another important contribution is provided by Vogelezang et al. [17]. The authors overviewed the application of the COSMIC method to modern software such as mobile and cloud applications. They also established four possible contexts in which different kinds of FURs can be identified, considering and measuring different data movements (see Fig. 3.4). Each context is an extension of the previous one, increasing the deepness in the cloud infrastructure:

1. The interactions are those between the functional user and the user interface service (UIS) running in the cloud environment and between user interface and some business processes.

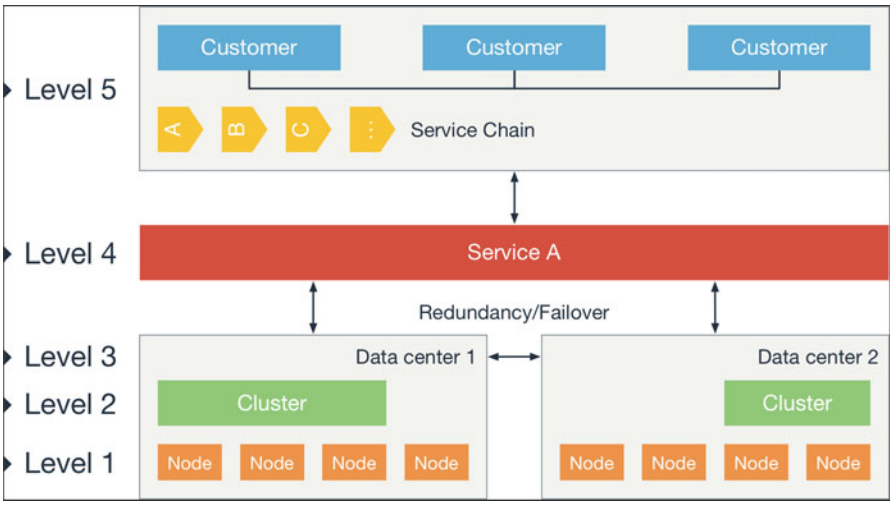


Fig. 3.3 Considered cloud system architecture

2. The FURs generated involve functional changes to hosted services on different machines in the same cluster.
3. The FURs include intra-component data movements between hosted services on different clusters.
4. An integration service allows the direct interaction between all the services in the application stack, using the integration service.

3.3.2 Measuring the Interaction Between SOA Services in the Cloud Environments

Distributed applications in cloud environments can be considered as a particular form of SOA software in which cloud-specific services can occur. Thus, the official guideline for sizing SOA with COSMIC can be applied [18].

An SOA-based software is designed following a specific pattern in which application components provide services to other components, exchanging data through a communication protocol (e.g., messaging queue), typically over a network. Even though sizing service-oriented software with Function Point Analysis fails when reconstructing or mapping the Functional User Requirements, the COSMIC method defines the concept of 'layers' that perfectly matches the SOA-based software sizing, without needing to adapt the method in any particular way.

The organisation for the Advancement of Structured Information Standards (OASIS) defines SOA as 'a paradigm for organising and utilising distributed

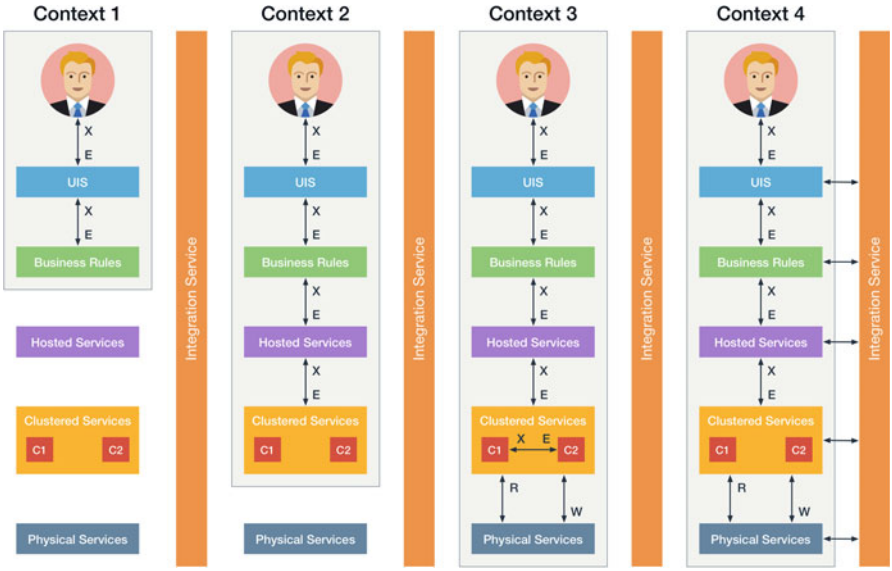


Fig. 3.4 Example of data movements in cloud software

capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations' [19].

Services are accessible to each other by means of a public interface, in the form of application program interfaces (APIs) often through simple Create/Read/Update/Delete (CRUD) operations. Not only do the APIs allow the services to communicate with each other but also they make the service independent in terms of development and execution. Indeed, the calling software does not have or need to know anything about how a service actually performs its tasks. This eases the independent work of parallel development teams and the inclusions of third-party components in SOAs. APIs are also useful for COSMIC sizing since they are independent of both involved technologies and implementations. Usually, APIs strictly follow the definition of FURs of the distributed application, being available from the early stages of the project.

Table 3.1 shows the classification given by the COSMIC guideline for SOA software [19].

The 'application services' provide specific business operations. They implement the features that characterise the distributed application; thus the most of the FURs are related to them. Each functionality can be invoked using an API in the form of network messages.

The role of 'orchestration services' is to call and control other services, often in an automatic way. An example of cloud orchestration service is the load balancer: it probes other services in the cloud system and checks the network and resource

Table 3.1 COSMIC guideline classification of service

Term used	Alternative terms	Description
Application service	Business service, entity service	Provides business functionality of an application
Orchestration service	Process service, task service	Controls ('orchestrate') application services to implement a (business) process
Intermediary service	Internal service, mediation service	Ensures independence of service requestors and service vendors
Utility service	Public service, software service	Provides common functionality (business or non-business) independent of, but made available to, any other applications

usage. It can scale the cloud system, by means of service instance replicas, to guarantee a balancing of load between the services and their consumers' satisfaction. Usually, the load balancer communicates with other services and, at the same time, with the cloud vendor, through the exposed API, being able to request the allocation of new resources or destruction of useless ones.

The communication management is delegated to the 'intermediary services' (i.e., message brokers) that interconnect a requestor's message with one or more application services. The intermediary services oversee controlling request and response messages, translating the language of messages and dealing with exception situations. Besides enabling service 'requestors' to communicate with service 'providers', another purpose of intermediary services is to ensure the independence between the two actors. This ensures flexibility allowing service vendor to change without needing to change service requestors and vice versa. This allows a high level of modularisation of the distributed application. A good example of intermediary service is a message queuing protocol: it ensures that the sent messages reach their destination, and it maintains a copy of messages until the requestors receive a confirmation or in the case of problems.

The 'utility services' provide functionality independently from other applications or services. For instance, a log service can be employed to monitor the application status to measure the service level agreement conditions. Also, they can be used to monitor the performance of cloud instance for statistics purposes. Another common example, especially in the cloud environment, is the discovery service: it allows services to reach and be reachable from other services. It is responsible for registering new services so that other services can query it and retrieve information data.

3.3.2.1 Measurement Strategy Phase

As for any target of a COSMIC measurement, for the SOA it is needed to specify the following elements.

In the case of measurement at the service level, usually the purpose of measurement is the estimation of the effort needed for the development or modification of

the system. The measurement can be performed with different scopes. The guideline for SOA [18] distinguishes between the cases where the purpose is to measure a piece of software 'using' multiple SOA services and when the purpose is to measure a 'collection' of multiple SOA services. For the first cases, the scope should be defined without considering internal data movements. In the latter cases, the size of the distributed application is equal to the sum of the sizes of single services.

What makes the COSMIC method particularly suitable for SOA services is the fact that it allows for the size measurement of multilayer applications. A layer, as defined by COSMIC, provides a set of services, which can be utilised by the software in other layers and can be part of a structure either hierarchical or bidirectional. In particular, in SOA the orchestration services can call application services but not vice versa establishing a hierarchical relationship. Intermediary services can be called by application service and vice versa, and both orchestration and application services can call utility services.

If the services in different layers need to be measured, the measurement scopes must be different. Figure 3.5 shows that the definition of what is considered as 'layer' depends on the 'view' of the software architecture. If the purpose is to measure the size of application A 'as a whole' as in (a), the measurement scope is the whole of application A as a single layer. If application A has been built according to the 'three-layer' architecture, the purpose is to measure the three components separately in view (b). In the case of SOA, the measurement scope must be defined separately for each SOA component.

Figure 3.6 shows an example of relationships between some SOA services [3]. The possible data movements between services are in the form of COSMIC Entry and Exit movements. COSMIC boundaries are set between a service and another. In the cloud context, also a persistent storage can be expected if the measurement scope includes the interaction between the service and the data storage on the same virtual instance where the service is running, thus within the boundary of the calling service. In this case, the data movements are expressed in terms of COSMIC Write and Read movements. Otherwise, if the storage is provided as a separate data service (e.g., a database service), another boundary involving Entry and Exit movements is needed.

The functional users of SOA services depend on the scope of the measurement and the purpose of the measurement. For instance, the functional user of an orchestration service is the application service that calls it. In the case of application services, its functional users can be the end users or other kinds of services that call it. For an intermediary service, the functional users are the application services that call it and the application and utility services that are called by it.

Let us assume to have a distributed application, defined with an SOA and composed of several services: the application itself, a log utility, a database and a load balance services. From the application service perspective, its functional users are all the other services. The log utility service interacts only with the application service in the same way for the database service. The load balance service, instead, can interact both with all the services of the SOA architecture but also with the

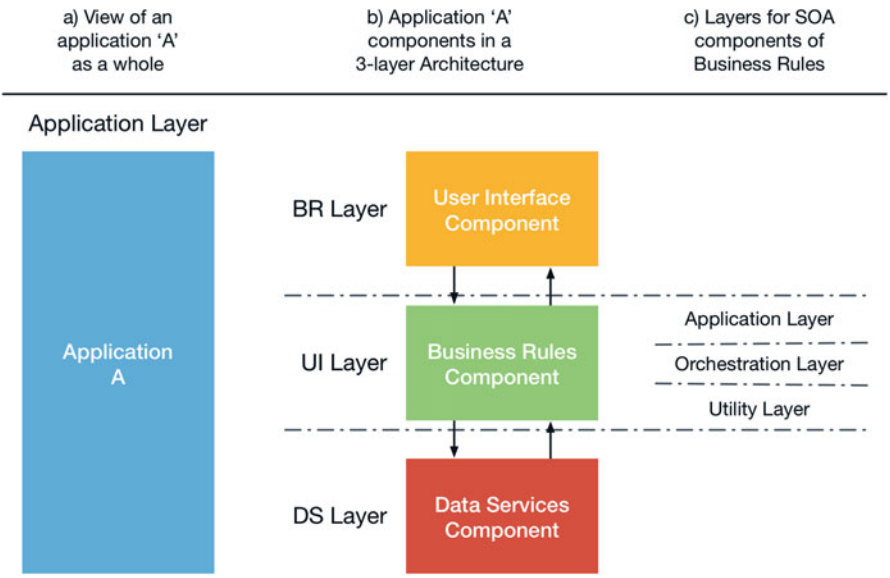


Fig. 3.5 Three views of the layers of an application

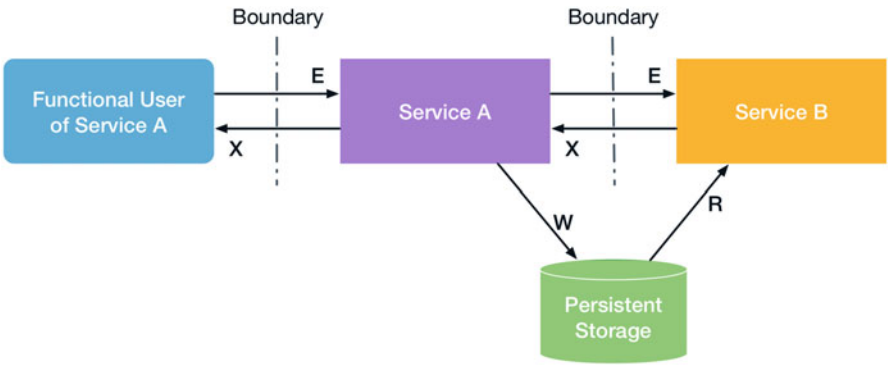


Fig. 3.6 Data movements between SOA services

cloud vendor service aiming, at the same time, to monitor the status of the resources in the system and demanding for creating/destroying resources to the cloud vendor.

3.3.2.2 Mapping and Measurement Phase

An FUR of a service may be restricted to define the 'capability' it provides for any service requestor, 'how' to request the capability and the 'form and content' of the request and reply messages. Some requirements that usually are considered as non-functional, in the case of SOA services, may be implemented directly as

software and therefore are part of the software. For instance, the security requirements need to follow precise protocols (e.g., OAuth) and the protocols being implemented in the software itself. If the protocol is in the scope of the measurement, it requires being measured as well.

The COSMIC method considers that unique events give rise to one or more functional processes whose role is to respond to the events. The first step for the measurement is to identify these functional processes and events. In the field of SOA, there are no standards about considering whether the concepts of 'service' and 'functional process' coincide; thus it is not excluded that one service may lead to multiple functional processes.

As mentioned before, the communication between services consists of messages exchanged, and therefore developing a service always involves developing the request/reply mechanism. The exchange between components may be 'synchronous' if the requesting service waits for the response before continuing its task. As an example of cloud service, it may be a web server that queries a database service to retrieve information. It can also be 'asynchronous' if the requestor functional process does not wait for the response message. For instance, an application service may ask for a time-consuming task to another service, without blocking its task. Once the latter finishes its job, the result can go back to the original requestor. In terms of COSMIC data movements, the main difference between the two kinds of services is that with the asynchronous mode, the arrival of a response message needs to be considered as another event triggering a separate functional process in the requesting software.

Another important thing that a COSMIC measure must deal with when measuring SOA services is the error message management. Technically speaking, in case a requestor calls another service and any issue occurs, the response message is replaced by the error data itself. In terms of COSMIC data movements, an Exit only is considered in any case. Nevertheless, if a confirmation/error message is notified to a human functional user or another service, an additional Exit must be considered.

3.4 Conclusions and Future Research Directions

It is widely recognised that the competitiveness of software companies greatly depends on the ability of their project managers to carry out a reliable and accurate software size estimation. Among the approaches proposed to size software FSM methods are widely applied in the industry since size estimation can be obtained early, based on the functionality provided to the users. In this chapter, we have analysed and discussed the main aspects of the use of the COSMIC method to measure distributed applications in cloud environments. In the discussion, we have considered the three distinct provision models of the cloud computing stack: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Furthermore, we have analysed specific concepts for distributed

applications such as orchestrations and load balancing of several components that act together to realise the required functionality and to ensure critical non-functional requirements (e.g., scalability, reliability).

It is clear that the effort for a software project and its related cost depend on both functional and non-functional aspects. On the basis of this consideration, in the last years, some approaches have been defined to measure the non-functional requirements of software systems to complement the information given by FSM methods. One of the most interesting examples is the Software Non-functional Assessment Process (SNAP) [20] devised by the IFPUG community. The aim of IFPUG is to capture functional aspects through the use of FPA and the non-functional ones with SNAP. The SNAP model consists of four categories and 14 subcategories to measure the non-functional requirements. Non-functional requirements are mapped to the relevant subcategories, and each subcategory is sized, and the size of a requirement is the sum of the sizes of its subcategories. These sizes are then summed to give the measure of the non-functional size of the software application. At the present, no empirical study demonstrating the effectiveness of SNAP is reported in the literature. Moreover, one of the main challenges is to understand which non-functional requirements do not give rise to functional components that are measured by FSM methods. In particular, there is the need to understand these aspects for the cloud environment also providing a specific SNAP approach. It is our intention to fill this gap as future work, in order to give a measure of the non-functional requirements of distributed applications.

Furthermore, empirical studies, possibly in the context of software companies, should be carried out measuring distributed applications, applying both COSMIC and SNAP and assessing the predictive accuracy of the built effort estimation models.

References

1. ISO (2007) ISO/IEC 14143–1:2007: information technology – software measurement – functional size measurement – part 1: definition of concepts
2. Albrecht AJ (1979) Measuring application development productivity. In: Joint SHAREGUIDEIBM application development symposium. pp 83–92
3. Abran A, Baklizsky D, Desharnais J-M, Fagg P, Gencel C, Symons C, Ramasubramani JK, Lesterhuis A, Londeix B, Nagano S-I, Santillo L, Soubra H, Trudel S, Villavicencio M, Vogelezang F, Woodward C (2015) The COSMIC functional size measurement method, measurement manual
4. Di Martino S, Ferrucci F, Gravino C, Sarro F (2016) Web effort estimation: function point analysis vs. COSMIC. Inf Softw Technol 72:90–109. doi:10.1016/j.infsof.2015.12.001
5. van Heeringen H, van Gorp E (2014) Measure the functional size of a mobile App: using the COSMIC functional size measurement method. In: Joint conference international workshop software measurement and the international conference software process and product measurement. IWSM-MENSURA. IEEE, pp 11–16

6. Ferrucci F, Gravino C,Salza P, Sarro F (2015) Investigating functional and code size measures for mobile applications: a replicated study. In: International conference prod.-Focus. Software process improvement. PROFES. pp 271–287
7. Nitze A, Schmietendorf A (2014) An analogy-based effort estimation approach for mobile application development projects. In: Joint conference international workshop software measurement and the international conference software process and product measurement. IWSM-MENSURA, pp 99–103
8. Zuse H (1997) A framework of software measurement. Walter de Gruyter & Co., Berlin
9. Trendowicz A, Jeferry R (2014) Software project effort estimation. Foundations and best practice guidelines for success. Springer, Cham
10. Albrecht AJ, Gaffney JE (1983) Software function, source lines of code, and development effort prediction: a software science validation. IEEE Trans Softw Eng 9:639–648
11. Conte SD, Dunsmore HE, Shen VY (1986) Software engineering metrics and models. Benjamin-Cummings Publishing Co., Inc., Menlo Park
12. Fenton NE (1991) Software metrics: a rigorous approach. Chapman and Hall, London
13. Gencel Ç, Demirörs O (2008) Functional size measurement revisited. ACM Trans Softw Eng Methodol. doi:10.1145/1363102.1363106
14. Abran A, Robillard PN (1994) Function points: a study of their measurement processes and scale transformations. J Syst Softw 25:171–184. doi:10.1016/0164-1212(94)90004-3
15. Kitchenham B (1997) Counterpoint: the problem with function points. IEEE Softw 14:29–31. doi:10.1109/MS.1997.582972
16. Schmietendorf A, Fiegler A, Wille C, Dumke RR, Neumann R (2013) COSMIC functional size measurement of cloud systems. In: Joint conference international workshop software measurement and the international conference software process and product measurement. IWSM-MENSURA. IEEE, pp 33–37
17. Vogelezang F, Ramasubramani JK, Arvamudhan S (2016) Estimation for mobile and cloud environments. Mod Softw Eng Methodol Mob Cloud Environ
18. Baklizky D, Lesterhuis A, Ozkan B, Symons C, Frank V (2015) Guideline for sizing service-oriented architecture software:1–31
19. Santillo L (2007) Seizing and sizing SOA applications with cosmic function points. Software Measurement European Forum SMEF
20. International Function Point Users Group (IFPUG) (2015) Software non-functional assessment process (SNAP), Assessment practices manual

Chapter 4
Characteristics of Large-Scale Defense Projects and the Dominance of Software and Software Project Management

Kadir Alpaslan Demir

4.1 Introduction

A defense system may be defined as *a system developed to increase the defense capabilities of a nation or an alliance*. Defense systems include weapon systems, command, control, communications, computers, intelligence, surveillance and reconnaissance (C4ISR) systems, warships, submarines, military aircrafts, helicopters, tanks, missile systems, satellite systems, unmanned systems, all other types of vehicles and systems developed for the military. While certain systems have civilian counterparts such as satellites, the main difference lies in the adherence to military standards mostly enforced by government acquisition regulations.

The global military expenditure exceeded $1.7 trillion in 2014 [1]. The USA led in the military expenditure with $609 billion in 2014. A significant portion of this spending goes to development and acquisition of defense systems. Defense projects are strategic, costly, complex, and time-consuming. Development of large-scale defense systems is challenging in many ways as evidenced by excessive cost and schedule overruns [2]. Surprisingly, scientific literature on defense systems is limited considering the importance and cost of these projects. The cost of defense systems is constantly increasing [4, 5]. Furthermore, the average time to deliver an initial capability to the warfighter is increasing [6], currently on average 1–2 months every year [6, 7]. Defense systems project management needs improvement [7–9]. Weapon system acquisitions are among high-risk areas in USA [7]. Overcoming the challenges of large-scale defense system acquisition and development projects starts with understanding the characteristics of these projects.

K.A. Demir (✉)
Department of Software Development, Turkish Naval Research Center Command, Istanbul, Turkey
e-mail: kadiralpaslandemir@gmail.com

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_4

Like many other types of systems, defense systems are evolving. In the past, defense systems were hardware-intensive. Today, they are software-intensive systems. In 1960, in an F-4, a military fighter aircraft, 8% of the functions were performed by software. In 2012, in an F-35 fighter aircraft, 90% of the functions are performed by software. Naturally, most of the development effort goes to software development in this defense project. Consequently, being a software-intensive system became an important characteristic of defense systems. As a result, the defense project management effort is actually becoming a software project management effort.

In this chapter, we first identify the current characteristics of large-scale defense systems. Then, we list the current characteristics of large-scale defense projects. The brief discussions related to each characteristic will reveal a heavy influence of software in these types of projects. Therefore, we finalize the chapter by emphasizing the importance of software project management in defense project management. This chapter is intended to help novice defense system project managers and other practitioners in understanding the specific aspects of these systems and their developments.

Let's clarify two terms for readers unfamiliar with acquisition processes of defense systems. There are two management offices in defense acquisitions. The program management office is established on the government side. This office manager, overseeing the defense system acquisition on behalf of the government, is generally called "Program Manager". On the contractor side, a project management office is established to manage the development of the defense system. The manager of this office is called "Project Manager". These terms will be used throughout the chapter.

Most references in this chapter originate from the USA for two reasons. First, the US government agencies generate a significant amount of reports and critics related to defense projects. Most of these reports are open to public and easily accessible. In addition, the USA is the major producer and consumer of defense software in the world and the leader in this aspect [10, 11]. As a result, most defense system development-related methodologies, processes, techniques, tools, regulations, and standards are first developed in this country. Many other countries adapt these developments. For example, UK's "Def Stan 07-8512 – Design Requirements for Weapons and Associated Systems [12]" refers to US "MIL-STD 498 – Software Development and Documentation [13]" for most software-related issues. Even though the infamous standard MIL-STD-498 is not supported anymore, it has wide use within defense community even in other countries [14].

4.2 Characteristics of Large-Scale Defense Systems

In this section, we identify the main characteristics of large-scale defense systems. Note that these characteristics are not unique to defense systems. Some of the civilian systems have a few of these characteristics. However, all or most of these characteristics exist in many large-scale defense systems. When most of these

Table 4.1 Characteristics of large-scale defense systems

Large-scale systems
Software-intensive systems
System of systems
Highly complex systems
High quality systems
Mission-critical systems
Safety-critical systems
Need for maintainability, supportability, and evolvability
Need for integration with legacy systems

characteristics exist in a system, then the system development becomes a real challenge. Table 4.1 lists the characteristics of large-scale defense systems.

4.2.1 Large-Scale Systems

By definition, large-scale defense projects are large in terms of scale. Managing large-scale projects is difficult for civilian projects, and it is even more challenging within defense context. This characteristic also leads to certain project characteristics such as high cost, need for long schedules, and involvement of a high number of stakeholders. The development of large-scale systems starts with adoption and disciplined execution of systems engineering principles and processes [22]. Large-scale projects require rigorous project management, good risk management, best systems engineering practices, experienced managers, skilled practitioners, well-crafted system architectures, a multi-aspect viewpoint, high skill in problem solving, etc.

Today, defense systems are software intensive [4, 5, 25]. In 1974, the F-16A had 135,000 source lines of code (SLOC). In 2012, operational and support software of F-35 consisted of 24 million SLOC [5]. The challenges of software development have dominated the challenges of defense system developments. As the project scale goes up, the rate of software project success falls dramatically [10, 16]. The cancellation rate for military software with a size of 1000 function points (FPs) is 10% [10]. The cancellation rate is 33% when the size of the military software reaches to 100,000 FPs [10]. Furthermore, productivity significantly lowers as the scale increases in military software [10].

In his seminal paper [15], David Parnas argued why Strategic Defense Initiative (SDI) system software will be untrustworthy. Most of the supporting arguments revolve around software-related problems in a large-scale system development effort. The scale is increasing in military systems [17]. Future defense needs, such as information dominance, will require systems of systems that will turn into "ultra-large-scale systems" primarily based on software [18]. The sheer scale in these projects will have unprecedented effects on many aspects of system development practices [18].

4.2.2 Software-Intensive Systems

Most defense systems incorporate significant amounts of software [24, 25, 120]. Additionally, almost all large-scale defense systems are software-intensive systems. Spruill states that "Now more than ever, software is the heart of our weapons systems" [56]. Today, the success of a defense system depends on the success of its system software [4, 5, 19, 57]. Mission-critical defense systems are plagued with software-related problems [58]. A military general states that "The B-52 lived and died on the quality of its sheet metal. Today, our aircraft will live and die on the quality of software" [5]. There are even firearm conceptual designs [59, 132] and patents [60, 61] that outline software modules for firearms.

There are programming languages designed for defense systems. For example, Jovial has its own military standard (MIL-STD-1589C) [65] and used in a number of programs [64], such as B-1B upgrade, and C-17 Globemaster III. Ada has common use in defense systems [19, 57]. C, C++ has some use, and Java is getting attention [19, 57, 66–68].

Later in the chapter, we devote a section discussing the software dominance in large-scale defense systems.

4.2.3 System of Systems

According to Defense Acquisition Guidebook [35], *a system of systems (SoS) is defined as a set or arrangement of systems that results from independent systems integrated into a larger system that delivers unique capabilities*. SoS is becoming an important means to increase military capabilities [36]. One of the trends in software and systems engineering is increasingly complex systems of systems [38].

Defense systems are envisioned to have a long life cycle since development of these systems is a long costly effort. For example, warships and fighters are expected to be in service for at least 20–30 years. If the life cycle is short, then the return on investment will be low. Thus, in most cases rather than replacing systems, enhancing their capability with upgrades and integrations to other or newer systems is preferred. Additionally, a defense system in service for some time is rather a proven system and more importantly a delivered system. Except a well-justified defense need, taking the risk and paying the cost of a new development effort for replacement are not easy decisions. Most current defense systems are expected to integrate with existing legacy systems [18]. They are also expected to be extendable and modifiable, so that they allow integration with future systems, hopefully with low cost. Therefore, most defense systems under development are either a SoS or a part of a SoS. The current projects carry the burden of both past and future project decisions. Thus, SoS-related issues are predominantly investigated in the defense sector [37]. Maier [131] identifies five principal characteristics of a SoS, as presented in Table 4.2.

Table 4.2 Principal characteristics of a system of systems

Characteristic	Discussion
Operational independence of the elements	Each element of the system of systems is able to operate independently and usefully
Managerial independence of the elements	Each component system is acquired seperately and each system continue its existence independent of the system of systems
Evolutionary development	The development of system of systems is in evolution as functions are added, removed, or modified
Emergent behavior	The system of systems has behaviors that do not exist in any of its components. These emergent behaviors are the reason for the existence of the system of systems
Geographic distribution	The elements of the system of systems are geographically dispersed

Since the article [39] by Admiral Owens, "Network-Centric Warfare (NCW)" [40–42] has become an important focus in defense community. In the last two decades, NCW becomes the strategic goal of many countries. The idea behind net-centric warfare is the networking of defense systems to accomplish more rapid and dynamic military capabilities [40–42]. There is an increasing emphasis on integration of systems to increase the warfare capability [43]. Furthermore, agile composition of services and systems to meet rapidly changing needs of warfighters will be a force multiplier in future wars. Emerging doctrines such as achieving full spectrum dominance [44, 45] require a successful set of SoS capable of effective NCW.

There are enterprise architecture frameworks (EAFs) [46–48] developed to support SoS developments. While EAFs such as Zachman framework [49], the Open Group's TOGAF (The Open Group, TOGAF version 9.1) [50], and US Federal Enterprise Architecture Framework (FEAF) [51] support the civilian domain, there are specific EAFs dedicated to defense SoS development. These frameworks include US DoD Architecture Framework (DODAF) [52], The British Ministry of Defence Architecture Framework (MODAF) [53], and NATO Architecture Framework (NAF) [54]. Object Management Group (OMG) has an attempt to unify these frameworks under a Unified Architecture Framework (UAF) [55]. The DODAF Version 2.0 views [52] are presented in Table 4.3.

4.2.4 Highly Complex Systems

Large-scale systems are inherently complex. Engineering complex systems requires a set of disciplined processes and systems engineering tools [70]. Complexity in defense projects is increasing [21]. The design of complex systems has considerable challenges [71].

Many military applications are real-time systems [19] and real-time systems are often complex systems [73]. Furthermore, weapon systems are safety-critical,

Table 4.3 DODAF version 2.0 views

All Viewpoint (AV)	*Services Viewpoint (SvcV)*
AV-1 Overview and summary information	SvcV-1 Services context description
AV-2 Integrated dictionary	SvcV-2 Services resource flow description
Capability Viewpoint (CV)	SvcV-3a Systems-services matrix
CV-1 Vision	SvcV-3b Services-services matrix
CV-2 Capability taxonomy	SvcV-4 Services functionality description
CV-3 Capability phasing	SvcV-5 Operational activity to services traceability matrix
CV-4 Capability dependencies	SvcV-6 Services resource flow matrix
CV-5 Capability to organizational development mapping	SvcV-7 Services measures matrix
CV-6 Capability to operational activities mapping	SvcV-8 Services evolution description
CV-7 Capability to services mapping	SvcV-9 Services technology and skills forecast
Data and Information Viewpoint (DIV)	SvcV-10a Services rules model
DIV-1 Conceptual data model	SvcV-10b Services state transition description
DIV-2 Logical data model	SvcV-10c Services event-trace description
DIV-3 Physical data model	*Standards Viewpoint (StdV)*
Operational Viewpoint (OV)	StdV-1 Standards profile
OV-1 High-level operational concept graphic	StdV-2 Standards forecast
OV-2 Operational resource flow description	*Systems Viewpoint (SV)*
OV-3 Operational resource flow matrix	SV-1 Systems interface description
OV-4 Organizational relationships chart	SV-2 Systems resource flow description
OV-5a Operational activity decomposition tree	SV-3 Systems-systems matrix
OV-5b Operational activity model	SV-4 Systems functionality description
OV-6a Operational rules model	SV-5a Operational activity to systems function traceability matrix
OV-6b State transition description	SV-5b Operational activity to systems traceability matrix
OV-6c Event-trace description	SV-6 Systems resource flow matrix
Project Viewpoint (PV)	SV-7 Systems measures matrix
PV-1 Project portfolio relationships	SV-8 Systems evolution description
PV-2 Project timelines	SV-9 Systems technology and skills forecast
PV-3 Project to capability mapping	SV-10a Systems rules model
	SV-10b Systems state transition description
	SV-l0c Systems event-trace description

mission-critical, embedded, reactive, and real-time systems [19]. These characteristics increase the complexity of defense systems [58]. Today, in the development of F-35 aircraft, the complexity is overwhelming. Software is the major source of complexity, even projected to be the main reason if F-35 project fails [74]. Based on

an analysis of 54 C3I and IT acquisition programs, Guernsey [72] believes that inability to manage complexity was the root cause of all failures.

Defense software is generally more complex than civilian software [75]. This is a consequence of requirements to provide greater functionality and higher reliability than commercial systems [75]. In software, testing effort and defect density significantly increase with complexity.

4.2.5 High-Quality Systems

Defense systems should be high quality. Only by achieving high quality, we will be able to ensure that our defense systems are trustworthy, safe to use, and able to complete the missions without failure. These systems should be operationally feasible meaning that the system should perform as intended in an effective and efficient manner for as long as necessary [76]. Achieving operationally feasible systems requires adherence to design for reliability, maintainability, and usability [76]. These qualities cannot be added as a component to the system. They are engineered in from the start.

Defense software ranks near the top in terms of quality among various types of software [11]. High quality costs. However, quality-related problems cost a lot more. An analysis of 11 weapon system projects showed that quality-related problems led to "billions in cost overruns, years-long delays, and decreased capabilities for the warfighter" [77]. Quality problems with the Expeditionary Fighting Vehicle program cost 4 years schedule overrun at a cost of $750 million [77]. Weapon systems cannot be fielded with low quality. Therefore, quality-related problems should be eliminated before delivery. Good systems engineering processes should be employed in the prevention of quality problems [77].

As the functionality achieved by software in defense systems is increasing, software quality is becoming more important for defense community [62]. Software quality has become the main determining factor in defense system quality. While there are many different quality attributes, a list of most commonly used quality attributes is presented in Table 4.4.

4.2.6 Mission-Critical Systems

A mission-critical system is a type of system in which a failure may result in not achieving a critical goal, significant loss in terms of money, or trust in the system [19]. In the defense context, the failure of a mission-critical system may cause a mission failure or a deficiency in the defense capability temporarily or permanently. Most defense systems are mission-critical systems, and most mission-critical systems are safety-critical systems. Defense mission-critical systems are highly complex and require millions of line of software code to support them [58]. Therefore,

Table 4.4 A list of commonly used quality attributes

Reliability	Usability	Producibility
Performance	Interoperability	Adaptability
Fault tolerance	Extensibility	Dependability
Safety	Modifiability	Efficiency
Security	Portability	Effectiveness
Availability	Reusability	Evolvability
Testability	Robustness	Modularity
Maintainability	Scalability	Portability
Supportability	Recoverability	Affordability
Simplicity	Survivability	Resilience

software-related problems hinder defense system developments. Lack of management attention, ill-defined system requirements, and inadequate testing are among the top sources of problems associated with the development of mission-critical systems [58]. Achieving the right level of trust in a defense system is also important [78, 79]. For example, overreliance on a UAV system may result in the loss of the vehicle due to harsh conditions, and under reliance may result in mission failure [80]. Development of mission-critical defense systems requires a rigorous system development process ending with a high-quality product.

4.2.7 Safety-Critical Systems

A safety-critical system may be defined as *a system whose failure may cause injury or death to human beings* [57]. While certain defense systems such as weapon systems are specifically developed to inflict damage on the enemy, they should be safe for friendly forces. Developing safe systems is a challenge by itself, and developing safe weapon systems having destructive power takes the challenge to a higher level.

MIL-STD-882E requires a system safety approach for managing hazards as an integral part of the systems engineering process [96]. Having a safety perspective from the start; implementation of formal methods for safety assurance; good, simple, and easy to understand design; attention to high quality in all phases of development; and rigorous testing are among the techniques used in the development of safety-critical systems [81]. Furthermore, fault tree analysis (FTA), hazard and operability studies (HAZOP), and failure mode effects analysis (FMEA) are specific techniques used in developing these systems [81]. The UK MoD Standard DEF-STAN-00-56 [82] requires safety analysis on all defense systems. Managers should be aware that developing safety-critical systems require expertise. However, practitioners educated on the development of safety-critical systems are limited. Project managers are advised to get training on strategies and techniques to develop safe systems and have a safety perspective from the project start. An overview of safety standards is listed in [83].

4.2.8 Need for Maintainability, Supportability, and Evolvability

The importance of life-cycle management of systems was recognized by defense agencies a long time ago [108]. Large-scale defense systems generally have long life cycles [109]. Maintainability and supportability of defense systems over a long life cycle are important. DoD 5000.01 [95] states that "Acquisition programs shall be managed through the application of a systems engineering approach that optimizes total system performance and minimizes total ownership costs. A modular, open-systems approach shall be employed, where feasible." Having a "modular open systems approach" is the key to supportability [109] and expected to improve the acquisition process [128]. Using an open systems approach with open interfaces, the design process is focused on lowering the entire life-cycle costs of weapon systems [109].

Most defense systems will evolve over time [110]. Therefore, it is quite likely that at some point in the future, the system will be modified/extended and integrated with other systems. A well-designed system architecture is essential for achieving required system qualities and a successful system evolution [111].

4.2.9 Need for Integration with Legacy Systems

As the cost to replace a defense system increase, many governments prefer the new defense systems (often system of systems) to be integrated with legacy systems to increase the warfighting capability [112]. As a result, the project managers are faced with the challenge of designing the new defense systems in a way that the system is able to integrate with a number of existing systems, some of which are legacy [18]. Even in defense systems that have no physical integration with a legacy system, there may likely be some outdated protocols (such as communication, data link, etc.) to be supported. Such requirements increase the complexity, and sometimes the new defense system design has to carry the burden of old design decisions. In addition, there is a high possibility that the documentation of legacy systems is either poor or lost in time if ever produced [113]. Without adequate information regarding the legacy system, integration problems may occur. Rigorous testing will be required. Performance problems may occur, and these problems may easily be attributed to the performance of the new system. Then the project managers face the burden of proofing that their systems are not the source of the problem. Another challenging issue may be the changes in the notions, terminology, or data representation. The warfare terminology evolves over time, and different meanings are attributed to old terminology. This evolution has to be managed, and the developers have to be aware of such warfare terminology evolution. Otherwise the integration with legacy systems will be more difficult and prone to errors.

Table 4.5 Characteristics of large-scale defense projects

Development under government acquisition regulations
Involvement of many of stakeholders
Long schedules
High cost
High risk
Security orientation
Slow development and low productivity
Process orientation
Adherence to many standards
Verification and validation orientation

4.3 Characteristics of Large-Scale Defense Projects

While there are common aspects of large-scale defense system projects such as scale, complexity, etc. with civilian projects of the same size, there are also unique aspects of large-scale defense projects such as the defense context and the influences of defense acquisition policies. The civilian projects sharing common aspects face similar challenges with defense projects. However, the defense context and the influences of acquisition policies increase the complexity and challenges to a higher level. Therefore, understanding these characteristics and their implications is crucial in overcoming the challenges in these projects. Table 4.5 lists the characteristics of large-scale defense projects.

4.3.1 Development Under Government Acquisition Regulations

Defense acquisition processes are subject to heavy government regulations and bureaucracy. The management of these acquisitions is inefficient, cumbersome, and bureaucratic [9, 86]. Defense acquisitions need improvement [92].There are reform attempts at the legislation level [84]; however many reform attempts have failed [85]. The armed services strategy of "low cost, high innovation" is in fact far from reality [85]. Jones [10] advises that military contracting should adopt some of the modern contracting practices of the civilian contracting.

The US weapon system acquisition milestone decision process employs A, B, and C milestones [9]. In each milestone, the program management office compiles the necessary information for the milestone decision authority. Based on a survey [86] of 24 weapon acquisition programs, on average, program management offices spent over 2 years completing up to 49 information requirements for their most recent milestone decision. The reason for this long period is the involvement of a high number of stakeholders that participate in preparing and reviewing the documentation [9]. This is understandable when the cost and role of these systems

are considered. However, most managers think that these reviews add value to only 10% of the documentation [9].

Today, defense systems are software-intensive systems; however current systems engineering and acquisition practices and program managers still rely on historical hardware engineering and acquisition legacy [87]. There is a need for change in perspectives, processes, and capabilities to shift the emphasis from hardware to software [5]. Program managers should be trained to improve software skills [120].

Government policies on the limitations [93] of defense equipment suppliers and various contractors bring extra challenges [88]. In addition, import and export of defense systems are subject to both national and international laws, regulations, and agreements.

In conclusion, dealing with government regulations is quite challenging [19].

4.3.2 Involvement of Many Stakeholders

Large-scale defense projects inherently involve a high number of stakeholders. The main stakeholders include military personnel, armed forces, ministry of defenses, government acquisition agencies, other government agencies, defense contractors, and defense committees in congresses. While secondary stakeholders may not have a direct effect on the projects, they may have a certain level of positive or negative influence on the projects. Secondary stakeholders include intergovernmental organizations, for example, NATO, other defense contractors who did not get a piece from the contract, political actors such as lobbyists or activists, public nonprofit organizations, citizens, etc. Even some defense projects are developed in a multinational context [19]. For example, nine different countries participate in the F-35 Joint Strike Fighter (JSF) program. Naturally, as the number of stakeholders increase, the possibility of having stakeholders with conflicting interests increases. The military personnel and the armed forces would like to be equipped with the latest technologies and get the most out of the defense system and the acquisition program. These expectations drive up the costs. However, the government acquisition agencies and defense and budget committees in the national congresses would like to see that the costs are kept at a certain limit. The ministry of defenses will be in between depending on the political climate. On the other side, using political lobbyists or other mechanisms, other defense contractors will try to get a piece from the contract and try to influence other stakeholders. Herndon [20] states that management of large defense projects requires considerable bureaucratic political skills.

From the government point of view, national priorities, defense needs, national long-term strategic goals, national defense acquisition policies, national technology and innovation level, long- and short-term national R&D goals, and international relations favoring or limiting the defense acquisitions from certain countries are

among the concerns affecting the defense system acquisition decisions and projects [21].

From the main defense contractor view, there are other challenges. In some cases, the government may impose certain products or subcontractors on the main defense contractor. This limits the options for the main contractors and gives a certain leverage to the subcontractor. The main contractor may be challenged in managing the subcontractor due to the imposed contract. In many cases, defense projects are developed by a main contractor and many subcontractors. Certain subtle challenges may arise unless the main contractor and all subcontractors do not have the same motivation for the project. For example, while some contractors may have a strategy to establish long-term relations with the government, some other contractors may see the project as a one-time business opportunity. The defense contractors who would like to have long-term relations with the government agencies will be likely to pay more attention to quality. However, the contractors who are in the project as a one-time deal will pay more attention to profit maximization at the cost of quality. These motivational differences will affect the defense system development process quality and lead to conflicts between contractors. These challenges will be subtle and will occur in the background leading to many not-so-obvious problems.

There are also government agencies that oversee the acquisition programs. The US Government Accounting Office [129] conducts annual assessment of Department of Defense's (DoD) major defense acquisition programs.

As systems become large, they are developed by a composition of dozens of companies bound by contracts at different levels [23]. These companies, forming a complex organization, may be distributed over a geographic area. There are a number of challenges [23] such as motivational differences, cultural differences, contract management, performance variations, language barriers, trust issues, information sharing, political and social connections, and geographical dispersions. Consequently, program managers on the government side and project managers on the contractor side should be aware of these challenges. There are no easy solutions to most problems related to having many stakeholders on the project.

4.3.3 Long Schedules

A defense system usually takes 5–10 years to accomplish full delivery [24]. Development of a large-scale defense system may take a decade or more [2]. Even reaching a final decision on a military software contract may result in a delay of 6–18 months [10, 11]. Therefore, a defense project is already late from the start. In defense projects, sometimes on-time delivery may be more important than achieving the projected cost. A defense system is developed in response to a defense need to increase national defense capability. The delay in the delivery of a defense system may result in deficiency in the national defense capability.

1969 was the year the Indian government decided to build a national light combat aircraft (LCA), HAL Tejas, a light-weight, multirole fighter aircraft [32]. After a long R&D process, the first prototypes were completed in 2000s. In 2015, Indian Air Force gets first indigenously built LCA Tejas. A fighter aircraft is one of the most complex defense systems. This example shows that a successful project of building a fighter may take over 40 years. Such projects require long-term strategic planning and overcoming many difficulties along the way. Similarly, Joint Strike Fighter (JSF) aircraft program was born in 1993. Even though the USA has quite an experience in building fighters, JSF [33] development is still uncompleted.

Earned value management (EVM) is a performance-based management tool that can be used determine and monitor cost and schedule performance [26–29] . EVM is widely employed by DoD [119], and the use of EVM is required from managers [30]. The benefit of using EVM is high in especially large-scale projects with long schedules. Defense Acquisition University (DAU) [118], established in 1991, was created as a result of critical necessity of providing defense workforce with a career path and consistent training opportunity. Completion of related defense acquisition courses and getting necessary certificates are among requirements for the DoD personnel participating in defense acquisitions. EVM is one of the major topics taught in DAU.

Average monthly requirements volatility for military software is about 2% [34]. Theoretically, a 1-year-long military project has a change in one-quarter of its requirements. Thus, as the project schedule lengthens, the amount of change in the requirements is increasing adding significant challenges.

4.3.4 High Cost

Development of a large-scale defense system requires a significant amount of effort. Therefore, these projects are inevitably costly. The portfolio of major defense acquisition programs reached $1.4 trillion in the USA as of 2014 [6]. The development and sustainment costs of the famous F-35 program are estimated to be over $1 trillion over 50-year projected life [5]. Much of this cost can be attributed to the complexity and size of the software development involved [5]. High-cost defense projects are under close scrutiny of government agencies [6]. Program and project managers are required to provide regular or on-demand performance reports to governing bodies. Therefore, these projects are constantly under political pressure. Consequently, keeping the costs under control and achieving a good cost performance are among the priorities of project managers.

Defense systems include military equipment and components that are subject to harsh environmental and battle conditions. Electronic components have two types of packaging: commercial and military specification. Components adhering to military specifications are expensive. Furthermore, some components in defense systems require certifications. The size of the required specifications and

Table 4.6 Sources of high cost in defense acquisition programs

Government acquisition policies
Military-grade equipment and components
Testing required for battle-ready systems
High volumes of documentation
Training a large number of military personnel
Certifications
Necessity for high quality

documentation is three times larger than it is in the civilian sector [10, 11]. All these issues are among the factors increasing the costs.

The costs of defense projects are increasing [6], and the cost performance of major defense projects is noticeably poor [6]. Table 4.6 lists the sources of high cost in defense acquisition programs.

4.3.5 High Risk

There are many reports [3] highlighting the high risks in large-scale defense system projects. Defense software projects have the highest cancellation rates [10]. Jones [10, 16] report that as the project scale goes up, the success rate falls dramatically in defense software projects. When the defense software system size is 1000 function points, only 10% of the defense projects face cancellation. However, when the defense software system size reaches 100,000 function points, the defense project cancellation rate increases to 33%. Note that these numbers only report project cancellations, meaning that the system could not be delivered in one-third of large-scale defense software projects. In addition, most large-scale defense systems are delivered with less functionality than what is initially planned and with quality problems. A US Government Accounting Office (US GAO) report [7] on high-risk government projects published in 2015 highlights that "Many DOD programs are still falling short of cost, schedule, and performance expectations." US Government Accounting Office publishes this "high-risk list" report every 2 years since 1990. Major weapon acquisitions found its place in the high-risk list in every report [3].

We conducted a survey study [127] on challenges of software projects and identified that half of the IT and software development projects are challenged in scope management. Not having a clear scope at the beginning of the project increases the risks in a project. Furthermore, as the duration and the scale of the project increase, the likelihood of scope changes increases. As a result, large-scale defense system projects face the challenges resulting from scope change over a long period of time due to many reasons.

Large-scale defense system projects are among the top in high-risk project lists. Therefore, the defense software project managers should conduct rigorous risk management from the project start as advised by experienced software program managers [117].

4.3.6 Security Orientation

Defense systems must be trustworthy and resilient to cyber attacks [62, 130] of all kinds over the whole life cycle. As cyber threats increase, importance of trusted defense systems is gaining significant attention in recent years [89, 121]. R&D and development of defense systems are conducted in secure environments [90]. The personnel working in the development of defense systems are required to have necessary clearances [90]. They are subject to background checks. In many cases, foreign nationals are not allowed to participate in defense projects. The developers working on critical modules are required to be citizens [121]. Even in some cases, only citizens born in the country are employed. In addition, the workplaces are secure environments with only authorized personnel access. The buildings are secured, and extra measures are taken against any type of intelligence gathering. Some development environments are remote locations that are far away from residences or general population. As a result, all these issues result in a limited pool of system developers to be employed in defense projects. Therefore, project managers are challenged in hiring of system developers with necessary skills. Furthermore, acquisition of defense system components from trusted accredited suppliers is essential in providing uncompromised weapons and information systems [91]. Developing secure systems has its own set of challenges [19, 121]. Though it is quite costly, critical components should be evaluated using common criteria [122] or other certifications.

4.3.7 Slow Development and Low Productivity

Even though defense software projects have high quality, they rank last in terms of software productivity [11, 31]. The size of the required specifications and documentation is three times larger than it is in the civilian sector [10, 11]. Today, 90% of functions in an F-35 aircraft are projected to be performed in software [5], and the software challenges slow the system development [74].

In military projects, as the scale goes up, productivity lowers [10]. In defense software development, average function points per staff month is 7.69 for 1000 function points (FPs), 2.48 for 10,000 FPs, and 0.88 for 100,000 FPs. As the scale goes up, the probability of completing the project on time significantly decreases. When the military project size is 1000 FPs, 65% of the projects are completed on time; when the project size reaches to 100,000 FPs, only 30% of the projects are completed on time [10]. Furthermore, based on the same statistics, 33% of large-scale defense software projects face cancellation. One of the main reasons of low productivity is that the amount of testing needed in large-scale defense projects is high. Consequently, project managers and stakeholders of large-scale defense projects should be aware that these projects should never be taken lightly during the whole development cycle.

4.3.8 Process Orientation

The adaptation of waterfall and V models through a set of military standards enforced contractors to adapt certain processes. Defense contracts include work definitions, deliverables at specific times, and milestones such as preliminary design reviews and critical design reviews with representatives from military and government acquisition agencies. In most contracts, delivering a system engineering management plan (SEMP) [76] and test and evaluation master plan (TEMP) [76] in the early phases of the development is a requirement. All these contractual bindings form a framework for a development process. Furthermore, the inherent nature of engineering large-scale defense systems requires a phased set of disciplined processes [70]. Testing of defense systems can be quite challenging [102]. There are unique problems such as the availability of testing sites, and testing opportunities with actual fielded systems. Testing in this scale requires careful planning and adaptation of certain processes.

Capability Maturity Model Integration (CMMI) [103–105] is a maturity-oriented appraisal and a key enabler for acquiring increasingly complex systems [106]. CMMI Version 1.2 has been adapted by many DoD and defense contractor organizations [107]. While CMMI does not enforce a specific model, accomplishing the required set of practices at different maturity levels aims at improving overall processes. Awarding the contracts to contractors with CMMI level 3 was highly recommended in the past [120]; however this requirement is being relaxed today.

4.3.9 Adherence to Many Standards

Standardization is important in military and defense. Standardization helps in achieving a level of quality and interoperability in defense systems [94]. When the customer is the government, the contractors are obliged to comply with many standards, regulations, and policies [19]. Government contracts may dictate a long list of standards (e.g., for system development [124–126], system safety [96], and avionics [123]) some of which requiring rigorous systems engineering practices. Consequently, requirements enforcing standards help us in achieving high-quality systems [19]. However, adherence to standards drives up the cost. Additionally, military standards are so demanding that the productivity in developing defense systems software is lower than any other industry [31].

4.3.10 Verification and Validation Orientation

The government acquisition processes enforce the development of defense systems to be verification and validation oriented. At milestones [9] and many phases in the

development, the program management office requires specifications, design review reports, development models, simulation results, measurements and analysis reports, certifications, test procedures, test results, etc. [76]. System development models such as waterfall [97, 98] and V model [99] are verification- and validation-oriented models commonly used in defense projects [57]. The US DoD captured the waterfall approach in a software development standard, DOD-STD-2167A [101], later superseded by MIL-STD-498 [13]. German Ministry of Defense adapted the V model as a standard for German Federal Armed Forces [100]. While all these documents or reports contribute to the verification and validation of the system being developed, they also increase the cost and pressure the schedule. Delays in the delivery of these artifacts may be penalized and may lead to delays in various development activities.

4.4 The Dominance of Software and Software Project Management in Large-Scale Defense Projects

Today, almost all large-scale defense systems are software-intensive systems. For example, F-35 Lightning II aircraft, one of the solutions in the Joint Strike Fighter program, consists of more than 8 million lines of code [4, 74]. In this aircraft, 90% of the functionality is achieved via software. This clearly shows how software dominated this defense system. The software development in this defense program is one of the largest software development ever. When we consider that F-16A Block 1 aircraft consists of only 135 thousand lines of code in 1974 [4], the increasing use and influence of software in major defense systems become obvious. Officials fear that the software challenges could cause the F-35 Lightning II aircraft program fail [74]. Furthermore, reports indicate that the software challenges are slowing the project [74]. Spruill [56] discusses how software became the heart of weapon systems. Ferguson states that "Software is the hidden, invisible power in weapon systems" [132].

Figure 4.1 shows the software code size in sample major defense systems. This figure was developed before 2006. Note that F-35 aircraft software and the ground software was estimated to include 14 million lines of source code at the time. However, today the software in this program reached to 8 million lines of code for aircraft software and 24 million lines of code for operational and support software. The increasing use of software in defense systems brings an important challenge from the software engineering discipline. Estimating the size of complex software systems is challenging [64], and frequently the estimations are either optimistic or underestimated. Incorrect estimations lead to cost overruns and schedule pressure resulting in reduced delivered functionality and/or lower quality [69].

While the size of software is increasing in defense systems, what is more interesting is the percentage of functionality achieved by software in defense systems. Table 4.7 shows the increasing trend of software use in a sample defense system, military aircrafts [63].

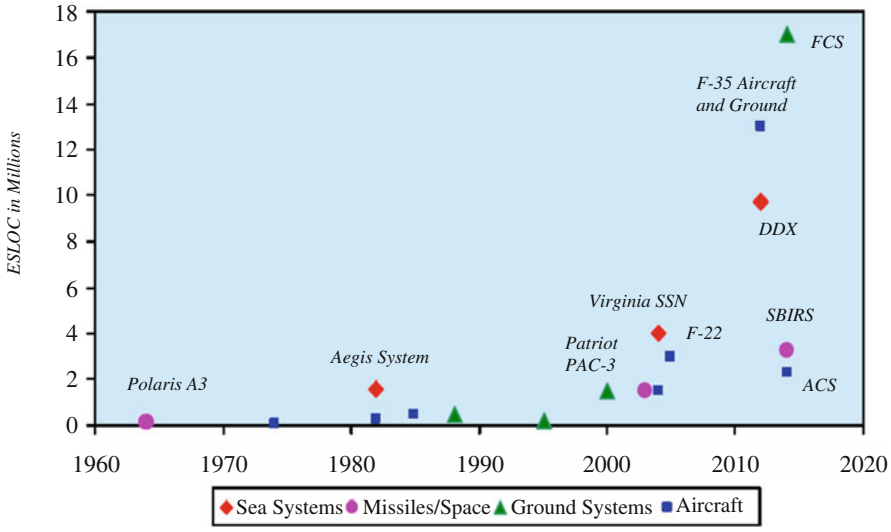


Fig. 4.1 Software source code size in sample major defense systems (Source: CARD Data, SEI, CSIS Analysis)

Table 4.7 System functionality requiring software

Defense system – military aircrafts	Year	% Functions performed by software
F-4	1960	8
A-7	1964	10
F-111	1970	20
F-15	1975	35
F-16	1982	45
B-2	1990	65
F-22	2000	80
F-35 Lightning II	2012	90

The productivity differences between civilian and defense projects are known for a long time. The US Assistant Secretary of the Navy established the Software Program Managers Network (SPMN) [114] in 1992. The goal of the initiative was "to identify proven industry and government software best practices and convey these practices to managers of large-scale Department of Defense (DoD) system acquisition programs." The initiative identified a set of best practices [115–117]. These 16 critical software practices are listed in Table 4.8.

The defense community was not successful in the adoption of civilian best practices. The defense community is quite conservative and therefore lags behind in adopting new technologies when compared to civilian industry [10]. Investigating the barriers of adoption and how to overcome these barriers are important research topics. Another topic might be investigating whether these best practices will produce similar results or not in the defense domain.

Table 4.8 Critical software practices for military software*

Adopt practices of continuous program risk management
Estimate costs and schedules based on empirical data
Use metrics to manage the program
Use earned value management and track earned value
Track defects against determined quality targets
Treat people as the most important resource
Adopt practices of full life cycle configuration management
Conduct requirements management rigorously
Use system-based software design
Guarantee data and database interoperability
Define and control all interfaces and related implementations
Design twice, but code once
Assess the risks and costs of reusable artifacts
Conduct requirements and design inspections rigorously
Adopt practices of continuous testing
Compile and smoke test the software system frequently

*Adopted from Refs. [10, 115, 116]

Capers Jones identified success and failure factors in military software [10]. Some of the best practices commonly observed in the military software domain are:

- Good process assessments and analysis
- Excellent research programs for best practices
- High success rates in large-scale applications (larger than 100,000 function points)
- Good research programs for reusability, computer-aided software engineering (CASE), and Ada programming language
- Good configuration control
- Good requirements traceability
- Good quality control
- Use of cost estimation tools

There are also some practices and issues leading to failure or poor performance in military software domain compared to civilian software. Some of them are:

- Slow adoption of functional metrics
- Inadequate productivity measurement technology
- Need for production of huge documentation
- Long schedules compared to all other types of software
- Low productivity compared to all other types of software
- High rates of challenges and mitigation in military contracts
- High growth of creeping user requirements
- Inadequate training and education of project managers and staff

Ferguson states, quoting an air force general, "The only thing you can do with an F-22 that does not require software is take a picture of it" [132]. Basically, the

development of defense systems became a software system development effort. Therefore, the defense project manager is in fact a software project manager. As a result, defense project managers should be well-trained in software project management.

4.5 Conclusion

In this study, we identify the characteristics of large-scale defense system projects. Understanding these characteristics will help project managers in preparing themselves for achieving better project results. Furthermore, as for the defense software researchers, identification of these characteristics forms a basis for further research. Even though there are studies identifying the best practices and investigating reasons of success and failure factors in these projects, the defense community still lacks coherent theories, proven methodologies, and development models for large-scale defense system developments. Identification of these characteristics, systematic analysis of challenges related to these characteristics, and investigating the ways to overcome these challenges will help the development of large-scale defense system development process models. Furthermore, this study establishes the heavy influence of software in defense systems as shown with many references to a body of current literature. Today, major defense projects become large-scale software development projects. As a result, the laws, theories, methodologies, practices, advantages, and limitations of software development are taking over defense projects. Therefore, defense project managers should be well-versed in software development and software project management. In addition, software engineering researchers should put more emphasis on large-scale defense projects, since these projects are becoming the examples of most costly, challenging, and risky software projects.

There are many system development process models such as waterfall model, spiral model, V model, evolutionary model, agile development models, etc. However, the success rates of defense projects clearly call for better models. The incremental commitment model [87] is an attempt for a process model that is claimed to be effective in various system developments including defense system developments. It is stated [87] that the milestones in the incremental commitment model are compatible with the milestones specified in the Defense Acquisition System [30]. However, the model has not been tested in actual defense projects.

As defense needs change and technology advances, defense systems will evolve over time. Naturally, the characteristics of defense systems will change. For example, in the past, the defense systems were not software intensive or system of systems. Today, defense systems are software intensive, and they are subject to the laws and challenges of current software engineering discipline. The characteristics identified here are the characteristics of current defense systems. Therefore, as defense systems evolve, researchers have to identify new or evolved characteristics.

Disclaimer and Acknowledgments The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any affiliated organization or government.

References

1. Stockholm International Peace Research Institute (SIPRI). Military Expenditure Database (2015) http://www.sipri.org/research/armaments/milex/milex_database/milex_database. Accessed 29 Sept 2015
2. Garrett RK, Anderson S, Baron NT, Moreland JD (2011) Managing the interstitials, a system of systems framework suited for the ballistic missile defense system. Syst Eng 14(1):87–109
3. U.S. Government Accountability Office (2009) High risk series, Report No: GAO-09-271
4. Hagen C, Sorenson J, Hurt S, Wall D (2012) Software: the brains behind U.S. Defense Systems, A.T. Kearney Inc. https://www.atkearney.com/documents/10192/247932/Software-The_Brains_Behind_US_Defense_Systems.pdf/69129873-eecc-4ddc-b798-c198a8ff1026. Accessed 29 Sept 2015
5. Hagen C, Sorensen J (2013) Delivering military systems affordably, Defense AT&L
6. U.S. Government Accountability Office (2015) Report to congressional committees: defense acquisitions – assessments of selected weapon programs, Report No: GAO-15-342SP http://www.gao.gov/assets/670/668986.pdf. Accessed 29 Sept 2015
7. U.S. Government Accountability Office (2015) Report to Congressional Committees: high risk series, Report No: GAO-15-290. http://www.gao.gov/assets/670/668415.pdf. Accessed 29 Sept 2015
8. U.S. Government Accountability Office (1999) Performance and accountability series: major management challenges and program risks – department of defense, GAO/OCG-99-4. http://www.gao.gov/assets/200/199558.pdf. Accessed 29 Sept 2015
9. United States Government Accountability Office, Report to Congressional Committees (2015) Annual report: additional opportunities to reduce fragmentation, overlap, and duplication and achieve other financial benefits, Report No: GAO-15-404SP. http://gao.gov/assets/670/669613.pdf. Accessed 29 Sept 2015
10. Jones C (2000) Software assessments, benchmarks, and best practices. Addison-Wesley Longman Publishing Co., Inc
11. Jones C (2002) Defense software development in evolution, Crosstalk –J Def Softw Eng. http://www.crosstalkonline.org/storage/issue-archives/2002/200211/200211-Jones.pdf. Accessed 29 Sept 2015
12. MODUK – British Defense Standards, Def Stan 07–85 – Design Requirements for Weapons and Associated Systems
13. U.S. MIL-STD 498 (1994) Software development and documentation
14. Overview of U.S. MIL-STD 498. https://en.wikipedia.org/wiki/MIL-STD-498. Accessed 29 Sept 2015
15. Parnas DL (1985) Software aspects of strategic defense systems. Commun ACM 28 (12):1326–1335. doi:10.1145/214956.214961
16. Humphrey WS (2005) Why big software projects fail: the 12 key questions. Crosstalk – J Def Softw Eng. http://www.crosstalkonline.org/storage/issue-archives/2005/200503/200503-Humphrey.pdf. Accessed 29 Sept 2015
17. Northrop L (2013) Does scale really matter? Ultra-large-scale systems seven years after the study (keynote). In: Proceedings of 2013 35th international conference on software engineering (ICSE), 18–26 May 2013, San Francisco, CA, USA, pp 857–857. doi:10.1109/ICSE.2013.6606633. http://resources.sei.cmu.edu/asset_files/Presentation/2013_017_101_68602.pdf. Accessed 29 Sept 2015

18. Northrop L et al (2006) Ultra-large-scale systems: the software challenge of the future. Carnegie-Mellon university, Software Engineering Institute (SEI), Pittsburgh. http://www.sei.cmu.edu/library/assets/ULS_Book20062.pdf. Accessed 29 Sept 2015
19. Demir KA (2009) Challenges of weapon systems software development. Journal of Naval Science and Engineering 5(3):104–116, http://www.softwaresuccess.org/papers/2009_Demir_JNSE_Challenges_of_Weapon_Systems_SW_Dev.pdf. Accessed 19 Feb 2017
20. Herndon RL (1983) The Army's National Training Center: a case study in management of a large defense project. Army Military Personnel Center, Alexandria, VA. (M.S. Thesis) http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA129072. Accessed 27 Sept 2015
21. Astan G (2015) Factors effecting technology acquisition decisions in national defense projects. J Def Res Manag 6(1):97–102. http://journal.dresmara.ro/issues/volume6_issue1/13_astan.pdf. Accessed 27 Sept 2015
22. BKCASE Editorial Board (2015) The guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.4. R.D. Adcock (EIC). Hoboken: The Trustees of the Stevens Institute of Technology. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society. www.sebokwiki.org. Accessed 29 Sept 2015
23. Bartholomew R, Collins R (2009) Evaluating an immersive virtual environment for organizationally distributed software development. In: AIAA Infotech@ Aerospace Conference, 6–9 April, Seattle, Washington, USA
24. Goldin L, Matalon-Beck M, Lapid-Maoz J (2010) Reuse of requirements reduces time to market. In: Proceedings of 2010 IEEE international conference on Software Science, Technology and Engineering (SWSTE), pp 55–60. 15–16 June 2010, Herzlia, Israel. doi:10.1109/SwSTE.2010.17
25. Nelson M, Clark J, Spurlock MA (1999) Curing the software requirements and cost estimating blues, PM Magazine, November–December, pp 54–60
26. Fleming QW, Koppelman JM (2006) Earned Value Project Management, 3rd edn. Project Management Institute, June 30, 2006
27. Fleming QW, Koppelman JM (1998) Earned value project management a powerful tool for software projects. Crosstalk – J Def Softw Eng. http://www.crosstalkonline.org/storage/issue-archives/1998/199807/199807-Fleming.pdf. Accessed 29 Sept 2015
28. Tomasetti R, Cohe S, Buchholz M (2005) Earned value management – moving toward government-wide implementation. Acquisitions Directions Advisory
29. Project Management Institute (2013) A guide to the Project Management Body of Knowledge (PMBOK Guide), 5th edn. http://www.pmi.org/PMBOK-Guide-and-Standards.aspx. Accessed 29 Sept 2015
30. U.S. Department of Defense Instruction (DoDI) 5000.2 (2015) Operation of the defense acquisition system. http://acqnotes.com/wp-content/uploads/2014/09/DoD-Instruction-5000.02-Operations-of-the-Defense-Acquisition-System-7-Jan-2015.pdf. Accessed 29 Sept 2015
31. Jones C (1998) Project management tools and software failures and successes, Crosstalk – J Def Softw Eng. http://www.crosstalkonline.org/storage/issue-archives/1998/199807/199807-Jones.pdf. Accessed 27 Sept 2015
32. https://en.wikipedia.org/wiki/HAL_Tejas. Accessed 29 Sept 2015
33. https://en.wikipedia.org/wiki/Joint_Strike_Fighter_program. Accessed 29 Sept 2015
34. Jones C (2007) Estimating software costs, 2nd edn. McGraw-Hill
35. Defense Acquisition Guidebook (2013) https://acc.dau.mil/docs/dag_pdf/dag_complete.pdf. Accessed 27 Sept 2015
36. Office of the Deputy Under Secretary of Defense for Acquisition and Technology (2008) Systems and software engineering. systems engineering guide for systems of systems, Version 1.0. ODUSD (A&T) SSE, Washington, DC. http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf. Accessed 27 Sept 2015

37. Schonenborg RAC, Bieler T, Matthyssen A, Fijneman M (2010) System of systems architecture in ESA's concurrent design facility. Proc SECESA 2010:13–15
38. Boehm B (2006) Some future trends and implications for systems and software engineering processes. Syst Eng 9(1):1–19. http://dx.doi.org/10.1002/sys.20044
39. Owens WA (1996) The emerging US system-of-systems (No. 63). National Defense University, Institute for National Strategic Studies, Washington, DC. http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA394313. Accessed 27 Sept 2015
40. Alberts DS, Garstka JJ, Stein FP (2000) Network centric warfare: developing and leveraging information superiority. Assistant Secretary Of Defense (C3I/Command Control Research Program) Washington, DC
41. Alberts DS, Garstka JJ, Hayes RE, Signori DA (2001) Understanding information age warfare. Assistant Secretary Of Defense (C3I/Command Control Research Program) Washington, DC
42. Cebrowski AK, Garstka JJ (1998) Network-centric warfare: its origin and future. In: US Naval Institute Proceedings, vol 124, no 1, pp 28–35. http://mattcegelske.com/wp-content/uploads/2012/04/ncw_origin_future.pdf. Accessed 27 Sept 2015
43. Dahmann JS, Lane JA, Rebovich G (2008) Systems engineering for capabilities. Crosstalk – J Def Softw Eng. http://www.crosstalkonline.org/storage/issue-archives/2008/200811/200811-Dahmann.pdf. Accessed 27 Sept 2015
44. US Joint Vision 2010. http://www.dtic.mil/jv2010/jv2010.pdf. Accessed 28 Sept 2015
45. US Joint Vision 2020. http://www.dtic.mil/dtic/tr/fulltext/u2/a526044.pdf. Accessed 17 Feb 2017
46. Schekkerman J (2004) How to survive in the jungle of enterprise architecture frameworks: creating or choosing an enterprise architecture framework. Trafford Publishing
47. Reichwein A, Paredis CJ (2011) Overview of architecture frameworks and modeling languages for model-based systems engineering. In: Proceedings of ASME 2011 international design engineering technical conferences and computers and information in engineering conference, pp 1341–1349
48. Urbaczewski L, Mrdalj S (2006) A comparison of enterprise architecture frameworks. Issues Inf Syst 7(2):18–23
49. Zachman J (1987) A framework for information systems architecture. IBM Syst J 26 (3):276–292. doi:10.1147/sj.263.0276
50. The Open Group (2015) TOGAF version 9.1. https://www.opengroup.org/togaf/. Accessed 29 Sept 2015
51. U.S. Federal Enterprise Architecture Framework (FEAF) (2015). Federal Enterprise Architecture Framework Version 2. https://www.whitehouse.gov/omb/e-gov/fea. Accessed 29 Sept 2015
52. U.S. Department of Defense. The DoDAF Architecture Framework Version 2.02. http://dodcio.defense.gov/Library/DoDArchitectureFramework.aspx. Accessed 29 Sept 2015
53. The British Ministry of Defence Architecture Framework (MODAF). https://www.gov.uk/guidance/mod-architecture-framework. Accessed 4 July 2016
54. NATO Architecture Framework (NAF) Version 4.0. http://nafdocs.org/. Accessed 4 July 2016
55. Object Management Group (OMG) Unified Architecture Framework (UAF). http://blog.nomagic.com/unified-architecture-framework-uaf-new-page-updm/. Accessed 29 Sept 2015
56. Spruill N (2002) Now more than ever, software is the heart of our weapons systems, Crosstalk-The Journal of Defense Software Engineering 3. http://www.crosstalkonline.org/storage/issue-archives/2002/200201/200201-Spruill.pdf. Accessed 4 July 2016
57. Demir KA (2005) Analysis of TLCharts for weapon systems software development. Masters' Thesis, Naval Postgraduate School, Monterey, CA, USA, December 2005. http://calhoun.nps.edu/bitstream/handle/10945/1825/05Dec_Demir.pdf?sequence=1. Accessed 19 Feb 2017

58. U.S. Government Accountability Office (1992) Mission-critical systems – defense attempting to address major software challenges, GAO/IMTEC-93-13, December. http://www.gao.gov/assets/220/217352.pdf. Accessed 4 July 2016
59. Portnoi M, Shen CC (2013) Secure zones: an attribute-based encryption advisory system for safe firearms. In: Proceedings of 2013 IEEE conference on Communications and Network Security (CNS), pp 397–398. 14–16 Oct. 2013. National Harbor, MD, USA doi:10.1109/CNS.2013.6682746
60. Milde KF, Jr (2015) U.S. Patent No. 8,931,195. U.S. Patent and Trademark Office, Washington, DC
61. Dietel B (2014) U.S. Patent No. 8,756,850. U.S. Patent and Trademark Office, Washington, DC
62. Nielsen PD (2015) Software engineering and the persistent pursuit of software quality. J Def Softw Eng:4–9. http://www.crosstalkonline.org/storage/issue-archives/2015/201505/201505-Nielsen.pdf. Accessed 4 July 2016
63. USAF (1992) "Bold Strike" executive software course
64. Judas PA, Prokop LE (2011) A historical compilation of software metrics with applicability to NASA's Orion spacecraft flight software sizing. Innov Syst Softw Eng 7(3):161–170
65. MIL-STD-1589C. JOVIAL (J73). http://everyspec.com/MIL-STD/MIL-STD-1500-1599/download.php?spec=MIL-STD-1589C.014577.pdf. Accessed 29 Sept 2015
66. Henties T, Hunt JJ, Locke D, Nilsen K, Schoeberl M, Vitek J (2009) Java for safety-critical applications. In: 2nd international workshop on the certification of safety-critical software controlled systems (SafeCert 2009)
67. Nilsen K (2004). Using java for reusable embedded real-time component libraries. Crosstalk: J Def Softw Eng:13–18. http://www.crosstalkonline.org/storage/issue-archives/2004/200412/200412-Nilsen.pdf. Accessed 29 Sept 2015
68. Nilsen K (2007) Applying COTS Java benefits to mission-critical real-time software. Crosstalk: J Def Softw Eng:19–24. http://www.crosstalkonline.org/storage/issue-archives/2007/200706/200706-Nilsen.pdf. Accessed 29 Sept 2015
69. Jones C (2006) Social and technical reasons for software project failures. Crosstalk: J Def Softw Eng:4–9. http://www.crosstalkonline.org/storage/issue-archives/2006/200606/200606-Jones.pdf. Accessed 29 Sept 2015
70. Sommerer S, Guevara MD, Landis MA, Rizzuto JM, Sheppard JM, Grant CJ (2012) Systems-of-systems engineering in air and missile defense. J Hopkins APL Tech Dig 31(1):5–20. http://www.jhuapl.edu/techdigest/TD/td3101/31_01_Sommerer.pdf. Accessed 29 Sept 2015
71. Drusinsky D, Shing MT, Demir K (2005) Test-time, run-time, and simulation-time temporal assertions in RSP. In: Proceedings of the 16th IEEE International workshop on rapid system prototyping (RSP'05), 8–10 June 2005, Montreal, Canada, pp 105–110
72. Guernsey GG (2009) Integrated test and evaluation (T&E) management: The Information Mission Assessment Tool (IMAT) prototype. PhD dissertation, Union Institute and University, USA
73. Gomaa H (2000) Designing concurrent, distributed, and real-time applications with UML. Addison-Wesley, p 8
74. Shalal-Esa A (2012) Pentagon focused on resolving F-35 software issues. Online News from Reuters, 30 March. www.reuters.com/article/2012/03/30/lockheed-fighter-idUSL2E8EU8C420120330. Accessed 29 Sept 2015
75. Defense Science Board (2000) Report of the defense science board task force on defense software, November, pp. 11.
76. Blanchard BJ, Fabrycky WJ (1998) Systems engineering and analysis, 3rd edn. Prentice Hall International Series in Industrial & Systems Engineering. ISBN: 0131350471
77. United States Government Accountability Office (2008) Report to congressional committees: increased focus on requirements and oversight needed to improve DOD's acquisition environment and weapon system quality, GAO-08-294. http://www.gao.gov/assets/280/271830.pdf. Accessed 27 Sept 2015

78. Lenfestey A, Cring E, Colombi J (2009) Architecting human operator trust in automation for multiple unmanned aerial system (UAS) control. In: Proceedings of software engineering research and practice 2009.121–127. Las Vegas, Nevada, USA. 13–16 July
79. Cring E, Lenfestey A (2009) Architecting human operator trust in automation for multiple unmanned aerial system (UAS) control. Master's thesis, Air Force Institute of Technology, USA
80. Demir KA, Cicibas H, Arica N (2015) Unmanned aerial vehicle domain: areas of research. Defence Science Journa 65(4):319–329. doi:10.14429/dsj.65.8631, http://www.softwaresuccess.org/papers/2015_Demir_UAV_Research_Areas.pdf. Accessed 17 Feb 2017
81. Storey NR (1996) Safety critical computer systems. Addison-Wesley Longman Publishing Co., Inc
82. UK Ministry of Defence Standard Def Stan 00–56 Safety management requirements for defence systems
83. Lee SY, Wong WE, Gao R (2014) Software safety standards: evolution and lessons learned. In: Proceedings of 2014 international conference on Trustworthy Systems and their Applications (TSA), pp 44–50. 9–10 June 2014, Taichung. doi:10.1109/TSA.2014.16
84. United States, Weapon systems acquisition reform act of 2009. http://www.govtrack.us/congress/billtext.xpd?bill=s111-454. Accessed 27 Sept 2015
85. Steinbock D (2014) The challenges for America's defense innovation. The Information Technology and Innovation Foundation (ITIF). http://www2.itif.org/2014-defense-rd.pdf. Accessed 27 Sept 2015
86. United States Government Accountability Office (2015) Report to congressional committees: acquisition reform – DOD should streamline its decision-making process for weapon systems to reduce inefficiencies, Report No: GAO-15-192, http://www.gao.gov/assets/670/668629.pdf. Accessed 29 Sept 2015
87. Boehm B, Lane JA (2007) Using the incremental commitment model to integrate system acquisition, systems engineering, and software engineering. Crosstalk – J Def Softw Eng 19 (10):4–9
88. United States Government Accountability Office (2006) Managing the supplier base in the 21st century, Report No: GAO-06-533SP. http://www.gao.gov/assets/250/249592.pdf. Accessed 29 Sept 2015
89. United States Under Secretary of Defense for Acquisition, Technology, and Logistics And Assistant Secretary of Defense for Networks and Information Integration/DoD Chief Information Officer (2009) Report on Trusted Defense Systems in response to the National Defense Authorization Act for Fiscal Year 2009, December 22. http://www.acq.osd.mil/se/docs/TrustedSystems-Exec_Summ-wAddendum-wTitlePgNoteinPDF.pdf. Accessed 29 Sept 2015
90. Demir KA (2016) Strategic human resource management of government defense R&D organizations, CrossTalk J Def Softw Eng 29(2):24–30. www.crosstalkonline.org/storage/issue-archives/2016/201603/201603-Demir.pdf. Accessed 19 Feb 2017
91. United States Department of Defense Instruction (2012) DoDI, 5200.44. Protection of Mission Critical Functions to Achieve Trusted Systems and Networks (TSN). November 5. http://www.dtic.mil/whs/directives/corres/pdf/520044p.pdf. Accessed 27 Sept 2015
92. United States Government Accountability Office (2015) Defense acquisition process, military service chiefs' concerns reflect need to better define requirements before Programs Start Report No: GAO-15-469, http://www.gao.gov/assets/670/668629.pdf. Accessed 29 Sept 2015
93. United States Government Accountability Office (2014) Defense Contracting: DOD's Use of Class Justifications for Sole-Source Contracts, 16, Report No: GAO-14-427R DOD Class Justifications. http://www.gao.gov/assets/670/662579.pdf. Accessed 29 Sept 2015
94. U.S. Department of Defense Manual 4120.24 (2014) Defense Standardization Program (DSP) Procedures. http://www.dtic.mil/whs/directives/corres/pdf/412024m.pdf. Accessed 29 Sept 2015

95. U.S. Department of Defense Directive (DoDD) 5000.01 (2007) Operation of the Defense Acquisition System. DoD Directive 5000.1: DoD. http://www.acqnotes.com/Attachments/DoD%20Directive%205000.01.pdf. Accessed 20 Sept 2015
96. Department Of Defense Standard Practice, System Safety, MIL-STD-882E, May 11 (2012) https://acc.dau.mil/adl/en-US/683694/file/75173/MIL-STD-882E%20Final%202012-05-11.pdf. Accessed 20 Sept 2015
97. Royce W (1970) Managing the development of large software systems: concepts and techniques. In: Proceedings of IEEE WESCOM. IEEE Computer Society Press, Los Alamitos
98. Petersen K, Wohlin C, Baca D (2009) The waterfall model in large-scale development. In: Product-focused software process improvement. Springer, Berlin/Heidelberg, pp 386–400
99. Hirschberg M (2000) The V model, Crosstalk – J Def Softw Eng,. http://www.crosstalkonline.org/storage/issue-archives/2000/200006/200006-Hirschberg.pdf. Accessed 29 Sept 2015
100. German Directive 250 (1992) Software development standard for the German Federal Armed Forces, V-Model, Software Lifecycle Process Model
101. US Department of Defense Standard (1988) DOD-STD-2167A Defense Systems Software Development
102. United States Government Accountability Office (2009) Defense acquisitions – charting a course for improved missile defense testing, 25, Report No: GAO-09-403T. http://www.gao.gov/new.items/d09403t.pdf. Accessed 29 Sept 2015
103. Software Engineering Institute (2010) CMMI® for Acquisition (CMMI-ACQ) Version 1.3, Technical Report: CMU/SEI-2010-TR-032. http://repository.cmu.edu/cgi/viewcontent.cgi?article=1277&context=sei. Accessed 29 Sept 2015
104. Software Engineering Institute, Carnegie Mellon University (2010) CMMI for Development (CMMI-DEV) Version 1.3, 2010. Technical Report: CMU/SEI-2010-TR-033
105. Software Engineering Institute (2010) CMMI® for Services (CMMI-DEV), Version 1.3. Technical Report: CMU/SEI-2010-TR-032, November 2010. http://repository.cmu.edu/cgi/viewcontent.cgi?article=1279&context=sei. Accessed 29 Sept 2015
106. Barbour R (2006) CMMI DoD perspective. Presentation. http://resources.sei.cmu.edu/asset_files/presentation/2006_017_001_22727.pdf. Accessed 29 Sept 2015
107. Phillips M, Shrum S (2010) Process improvement for all: what to expect from CMMI Version 1.3. Crosstalk – J Def Softw Eng, http://www.crosstalkonline.org/storage/issue-archives/2010/201001/201001-Phillips.pdf. Accessed 27 Sept 2015
108. Lehman M (1980) Programs, life cycles, and laws of software evolution. Proc IEEE 68 (9):1060–1076. doi:10.1109/PROC.1980.11805
109. Larson AG, Banning CK, Leonard JF (2002) An open systems approach to supportability. WALCOFF Technologies Inc., Fairfax. http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA404574. Accessed 27 Sept 2015
110. Dahmann J, Baldwin K (2011) Implications of systems of systems on system design and engineering. In: Proceedings of 2011 6th international conference on system of systems engineering (SoSE), pp 131–136. 27–30 June. Albuquerque, NM, USA doi:10.1109/SYSOSE.2011.5966586
111. Demir KA (2015) Multi-view software architecture design: case study of a mission-critical defense system. Computer and Information Science 8(4):12–31. doi:10.5539/cis.v8n4p12, http://www.ccsenet.org/journal/index.php/cis/article/view/51025/28835. Accessed 19 Feb 2017
112. Dahmann J, Rebovich G, Lowry R, Lane J, Baldwin K (2011) An implementers' view of systems engineering for systems of systems. In: Proceedings of 2011 IEEE international systems conference (SysCon), pp 212–217. 4–7 April. Montreal, QC, Canada. doi:10.1109/SYSCON.2011.5929039
113. Victor B (2013) Revisiting legacy systems and legacy modernization from the industrial perspective. Masters' Thesis, University of Utrecht, Utrecht, the Netherlands
114. Software Program Managers Network (SPMN). https://acc.dau.mil/CommunityBrowser.aspx?id=219279&lang=en-US. Accessed 24 Feb 2017

115. Software Program Managers Network, 16 Critical Software Practices. https://acc.dau.mil/CommunityBrowser.aspx?id=191920. Accessed 24 Feb 2017
116. Evans M (2001) SPMN director identifies 16 critical software practices. CrossTalk – J Def Softw Eng. http://www.crosstalkonline.org/storage/issue-archives/2001/200103/200103-Evans.pdf. Accessed 29 Sept 2015
117. Software Acquisition Best Practices Initiative, Software Program Managers Network (SPMN) (1998), The Program Manager's Guide to Software Acquisition Best Practices Version 2.31
118. Defense Acquisition University (DAU). http://www.dau.mil/default.aspx. Accessed 29 Sept 2015
119. Tomasetti R, Cohe S, Buchholz M (2005) Earned value management moving toward Governmentwide implementation. Acquisition Directions. https://www.asigovernment.com/documents/adv05-08.pdf. Accessed 27 Sept 2015
120. Report of the Defense Science Board Task Force on Defense Software (2000) Defense Science Board, p 11. http://www.acq.osd.mil/dsb/reports/ADA385923.pdf. Accessed 27 Sept 2015
121. Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software (2007) Defense Science Board, pp 11. http://www.acq.osd.mil/dsb/reports/ADA486949.pdf. Accessed 29 Sept 2015
122. Common Criteria. http://www.commoncriteriaportal.org. Accessed 29 Sept 2015
123. DO-178C (2011) Software considerations in airborne systems and equipment certification
124. ISO/IEC/IEEE 15288–2008. Systems and software engineering – system life cycle processes. doi:10.1109/IEEESTD.2008.4475828, http://ieeexplore.ieee.org/servlet/opac?punumber=4475823. Accessed 29 Sept 2015
125. ISO/IEC/IEEE 15289–2011. Systems and software engineering – content of life-cycle information products (documentation). doi:10.1109/IEEESTD.2011.6104079, http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6104077. Accessed 29 Sept 2015
126. ISO/IEC/IEEE 15289–2015. International standard systems and software engineering – content of life-cycle information items (documentation). http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7270962. Accessed 29 Sept 2015
127. Demir KA (2009) A survey on challenges of software project management. In: Proceedings of software engineering research and practice (SERP 2009), July 13–16, 2009, Las Vegas, Nevada, USA, pp 579–585. http://www.softwaresuccess.org/papers/2009_Demir_SERP_Survey_On_Challenges_of_SW_Project_Mgmt.pdf . Accessed 19 Feb 2017
128. Rendon RG (2007) Using a modular open systems approach in defense acquisitions: Implications for the contracting process. In: Proceedings of IEEE international conference on system of systems engineering (SoSE'07). pp 1–6. 16–18 April. San Antonio, TX. doi:10.1109/SYSOSE.2007.4304231
129. U.S. Government Accounting Office. http://gao.gov/index.html. Accessed 29 Sept 2015
130. Defense Science Board (2013) Resilient military systems and the advanced cyber threat. http://www.acq.osd.mil/dsb/reports/ResilientMilitarySystems.CyberThreat.pdf. Accessed 29 Sept 2015
131. Maier MW (1996) Architecting principles for systems-of-systems. INCOSE Int Symp 6 (1):565–573
132. Ferguson J (2001) Crouching dragon, hidden software: software in DoD weapon systems. IEEE Softw 18(4):105. doi: 10.1109/MS.2001.936227

Chapter 5
Software Project Management as a Service (SPMaaS): Perspectives and Benefits

Muthu Ramachandran and Vikrant Chaugule

5.1 Introduction

Cloud computing has evolved to address the availability of computing resources which can be accessed from anywhere and anytime. In particular, computing hardware and software often gets outdated, and hence, it is wise to outsource computing resources and to manage their IT infrastructures outside of their company premises, which is more cost-effective than is the case at present. Applications can be leased (like pas-as-you-go service) rather than being purchased, and companies have increased their data centers due to demand (Amazon, Microsoft, and IBM) [1]. Cloud computing is heavily based on "software as a service" concept and needs high-speed web access. It provides services on demand utilizing resources more effectively within the cloud environment. The cloud architecture, its layers, and its composition of components and services need to be designed for scalability, security, and re-configurability as they support services and its agreements (e.g., service level agreements). In this scenario, the resource management of cloud computing is the key to achieving potential benefits.

Cloud computing, one of the greatest developments in the field of computing, has the ability to transform and change the work of an IT industry. It has definitely helped in making the way software can be offered more attractively and also changing the way hardware is purchased and designed. It has led to a complete

M. Ramachandran (✉)
School of Computing, Creative Technologies and Engineering, Leeds Beckett University, Leeds LS6 3QS, UK
e-mail: M.Ramachandran@leedsbeckett.ac.uk

V. Chaugule
Department of Computer Science and Engineering, National Institute of Technology, Surathkal, Karnataka 575025, India
e-mail: vikrant.chaugule@gmail.com

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_5

change especially that developers coming up with new Internet services need not require a large investment in hardware or the human resources to operate the hardware. The developers need not waste costly resources and face losses in case the product fails, and on the other hand, they do not need to worry about scalability if the idea turns out to be successful and popular. This versatility about resources, without paying a premium for vast scale, is phenomenal for the IT industry.

Cloud computing consists of both applications provided as services over the Internet and the hardware and systems software which provide such services in the data centers. The services themselves have long been referred to as software as a service (SaaS) [2]. Together, the hardware and software at the datacenter comprises the cloud. When this is made available in the form of a pay-as-you-go fashion, to the people, it is called a public cloud, whereas utility computing is the service being sold. If the cloud is owned privately by an organization only for storing their information and is not made available to the public, it is called a private cloud. Thus, the addition of SaaS and utility computing is called cloud computing. People can be users or providers of utility computing, or users or providers of SaaS. We would like to focus on SaaS project management with the help of improving the way services are provided such that it is more convenient for users to use and benefit from them.

Some of the benefits of using cloud computing are [3]:

- It leads to lowering of project costs. Since the model used for billing is pay as per usage, maintenance is reduced since infrastructure required is not purchased.
- A massive infrastructure is provided by all the cloud service providers, and therefore, managing large volumes of data has become a reality. The cloud can be scaled dynamically, and sudden workload spikes can be handled very efficiently.
- It is very flexible. With enterprises having to adapt and adjust very rapidly, delivery speed becomes very critical. Hence, more emphasis is given on getting applications to market very quickly.

With the emergence of cloud computing, the focus moves to the interface, that is, interface between the service consumers and service providers. Some areas like distributed services, risk assessment, procurement, and service negotiation will demand expertise from enterprises, but most of them are only modestly equipped to take care of them.

Cloud computing is based on web access; therefore, we need to design web applications which are designed for security. Hence, it is essential to design cloud applications as web service components based on well-proven software process, design methods, and techniques such as component-based software engineering (CBSE). Wand and Laszewski [4] define cloud computing as a set of network-enabled services which provides scalable, guaranteed QoS (quality of service), inexpensive computing platforms on demand, customizable (personalized), and all of which can be accessed in a simple and pervasive way. An overview of the different cloud computing paradigms is discussed and presented with definitions, business models, and technologies by Wand and Laszewski [4] and by many others.

Software components provide a good design rationale supporting various requirements of application developments, design flexibility, system composition, testability, reusability, and other design characteristics. Component-based designs are customizable, and interfaces can be designed supporting SLA (service level agreement). SLAs vary between service providers which need to be customized without much effort. This can only be achieved using a component which has been designed for flexible interface that links to a number of SLAs. Each SLAs and business rules can be represented as a set of interfaces that can be mapped onto knowledge-based database or a data server. This also allows the reuse of SLAs for any individual service providers. Some of the important characteristics of the cloud computing mentioned are:

- On-demand service
- Handling multi tenancy service requirements
- Resource grouping
- Efficient elasticity
- Measurable service delivery

Our earlier work described by Ramachandran [5] on component model for web services and service-oriented architecture (SOA), grid computing, and various other systems can become an integrated aspect of any cloud computing architectures and application design. We also need to understand the basic differences among SOA (service-oriented architecture), grid, and cloud computing. SOA is to offer services which are based on open standard Internet services and virtualization technology and have been running in a different environment, *grid* offers services from multiple environments and virtualization, and *cloud* combines both. We also need to identify a specific development process for capturing requirements, design and implementation strategies, security, and testing cloud applications. Cloud computing paradigm has lots to offer, but at the same time, we need to consider building a secured and resilient architecture and services that are reliable and trustworthy. In this chapter, a generic component model and a web service component model have been developed meeting the design demands for building cloud application architectures. In this research, we have also proposed architectural composition strategies which can be customized for various cloud services.

This chapter presents our work on software development process model for building cloud services as it is necessary to follow a systematic approach. The organization is as follows. Section 5.2 gives a detailed explanation on the software development process for Cloud computing, Sect. 5.3 talks about the service development process, Sect. 5.4 compares classical and cloud-based project management tools, Sect. 5.5 provides a critical evaluation of some existing project management tools, Sect. 5.6 discusses the integrated software development process, and Sect. 5.7 gives a detailed explanation on the service-oriented architecture. Conclusions are summarized in Sect. 5.8.

5.2 Software Development Process for Cloud Computing

In order to define a process model, it is useful to capture some of our thoughts on understanding the very nature of cloud characteristics and its type of services that aims to provide. Identifying characteristics of a service-oriented system is vital for designers such that they can select, design, and evaluate those characteristics that are applicable to their applications. Service-oriented computing (SoC) [6] involves integration of several disciplines and subject areas, and therefore, some of the characteristics will overlap. Some of the identified services and component characteristics are:

- Reusable web services and some other core services
- Enterprise integration services
- Dynamic binding and reconfigurable at run-time
- Granularity
- Publish, subscribe, and discover
- Open world where components must be able to connect and plug to third party software systems or components.
- Heterogeneity supporting cross-platform applications
- Reconfigurable
- Self-composable and self-recoverable
- Cloud infrastructure and resources management
- Autonomic framework
- Middleware
- QoS

This is illustrated in Fig. 5.1, which shows some of the above characteristics that are the key to developing software components. In the modern software development, characteristics such as open world where components can be customizable and connectable to third party systems and their components and heterogeneity are crucial to developing highly reusable web services that will apply across domains and services.

The main reason for presenting such characteristics is to understand the basis for service-oriented systems and hence providing good practice design guidelines. The next section looks at the distinct features and differences between services and components. Again these characteristics need to overlap as we are also interested in applying component-based development for service-oriented systems. In particular web services need to possess both services and component characteristics. After looking at service-oriented computing and the characteristics of SaaS systems in this section, we shall look at the service-oriented development process in detail in the next section.

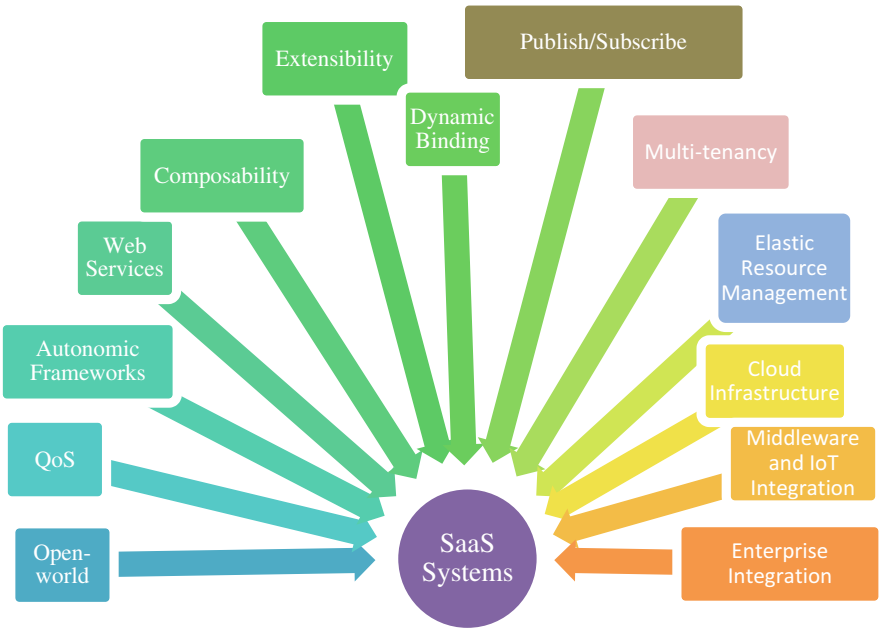


Fig. 5.1 Characteristics of software as a service systems (SaaS)

5.3 Service Development Process

The identification of service requirements [6] needs a new RE process and modelling techniques as it is highly dependent on multilevel enterprises across corporation. Identifying and knowing all requirements for all expected and even unexpected services is very hard. The idea in software as a service is to publish automatically new services whereby service agents can then be able to request and take advantage of required services for their customers. Figure 5.2 shows a development process model for service-oriented computing where initial requirements are captured based on enterprise-wide techniques and perhaps using domain analysis which should focus on a family of products and services. The second phase (RE services) involves identifying a set of requirements of system services. This process involves service modelling and service specification for which we can use any well-known techniques such as use case design and a template for software as a service (SaaS).

The third phase (categorizing services) involves classifying and distinguishing services into various categories such as enterprise integration services (services across corporations, departments, other business services); software services which represents core functionality of software systems; business logic services which represents business rules and its constraints; web services (a self-contained and web-enabled entity which provides services across businesses and customizable at

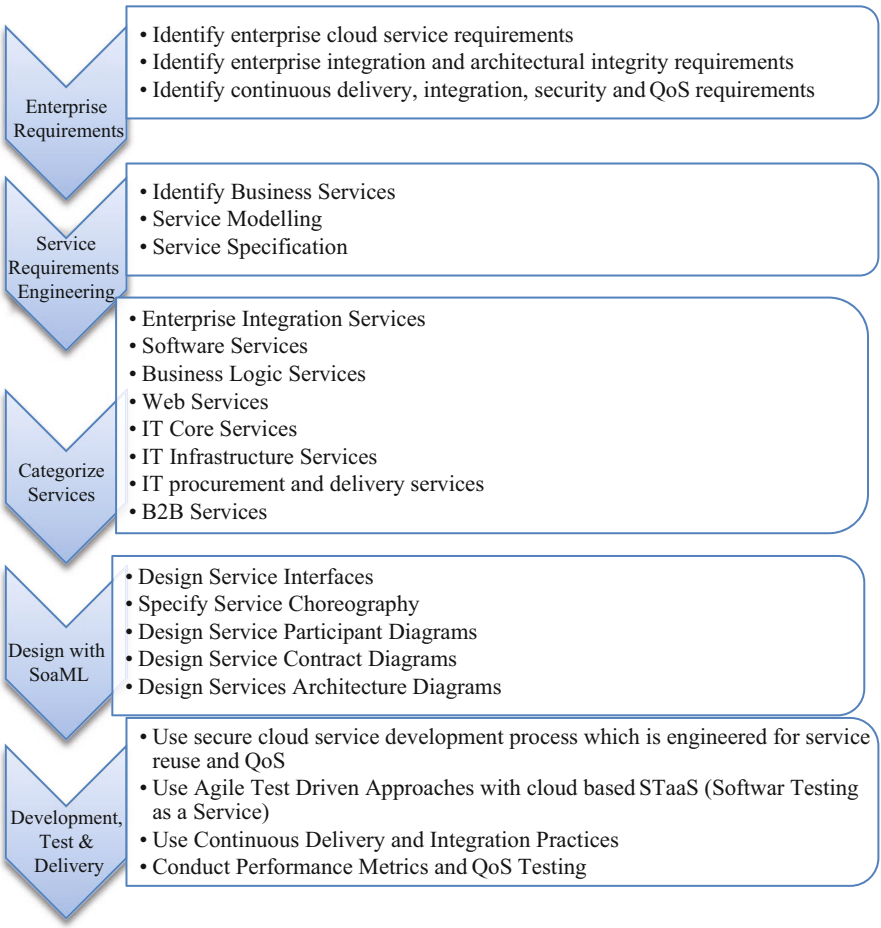


Fig. 5.2 Service-oriented software development process

run-time); IT core services which include resource management, help desk systems, IT infrastructure, and procurement; and delivery services, B2B and B2C services, data services, QoS services, middleware services, transaction management services, process integration services, re-configurability services, and grid services which include grid resource management and re-configurations. Service design stage has been proposed with designing services using OMG standard design notation known as SoaML which consists of a five-stage design:

- Design Service Interfaces: This offers services to other services through well-designed interfaces (the value provided), it allows design for service reuse, and it allows modelling of service specification.

- Specify service choreography which defines the interaction between the provider and consumer in completing a service, and this can be modelled using UML sequence diagrams.
- Design Service Participant Diagrams: The concept of a participant, in SoaML, represents certain party or component that provides a transaction through its interface to a consume service(s). Participants can be software components, organizations, system, or individuals. The participant design should be designed with SoaML participant diagrams which are similar to a UML component service with provider and require interfaces designed through the concept of a port. Service participant diagram allows for modeling primarily the participants that play role(s) in services architectures. It also presents the services provided and used by these participants.
- Design Service Contract Diagrams: As we have discussed, there are three approaches to specify a service: the above two interface-based approaches – simple interface and service interface – and, thirdly, through a service contract. Service contract defines the agreement between parties about how a service is to be provided and consumed. "Agreement" here refers to interfaces, choreography and any terms and conditions. Interacting participants MUST agree to the agreement in order for the service to be enacted. In SoaML, this is designed using service contract diagrams.
- Design Services Architecture Diagrams: This stage of the design offers features to express the complete list of services and their interactions. A service-oriented architecture, abbreviated as SOA, shows the participant roles that provide and consume services to fulfill certain purpose. In SoaML, this is represented as large globe with all interacting services connected.

We discuss in detail, in the last section of this chapter, SoaML design for SPMaaS. Based the above finding, we can propose a new paradigm for cloud application engineering as shown in Fig. 5.3. This illustration provides a relevant link to classical software engineering process.

As shown in Fig. 5.3, the requirements phase is linked to identifying cloud requirements which should in particular identify service requirements and relevant software security requirements so that cloud services are built with security rather than adding security batches after release. The design phase is linked to designing services for cloud environment and reuse as services are loosely coupled and have high potential for reuse. The code/implementation phase is linked to service development. Likewise testing and QA are related to cloud testing strategies and quality engineering. The next section discusses classical software project management activities and the need of cloud-based project management tools and highlights the advantages of cloud-based project management tools [11–17].

With increase in the use of service-oriented architecture, software projects and systems can get very complex. With the aim of managing this complexity, a number of SDLC models have been used:

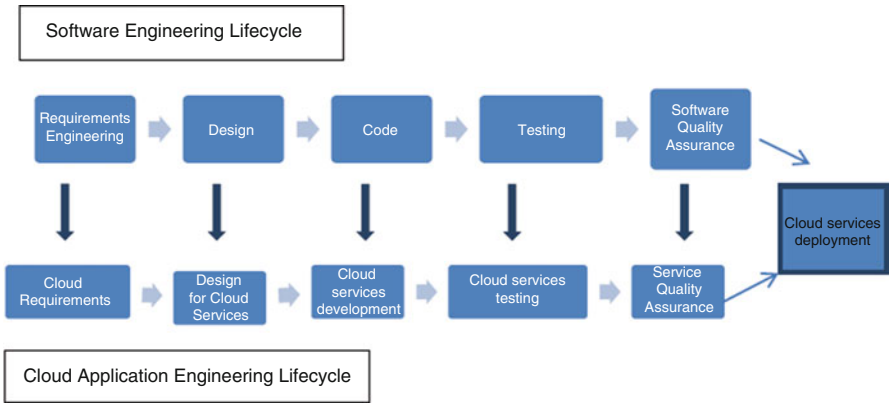


Fig. 5.3 Software engineering vs. cloud application engineering lifecycle

- Waterfall model
- Spiral model
- Rapid prototyping
- Agile method
- Incremental
- Synchronize and stabilize

The above mentioned models have been in existence for many years, and each of them has their own advantages. For instance, the waterfall model is simple to understand, use, and manage. High amount of risk analysis is done with the spiral model, and hence avoidance of risk is enhanced. The agile method allows regular adaptation to changing circumstances, and even late changes in requirements are welcomed. The incremental model provides benefits like easier testing and debugging during smaller iterations and also lowers initial delivery cost. Though these models have their own advantages, there are many issues which exist:

- Fulfilling compatibility criteria of numerous services from various vendors
- Sufficient bout of resources and accordingly manage them
- Lack of required coordination between client and provider to provide what was required
- Managing responsiveness and streamline changes as requested
- Deviation from anticipated product
- Difficult to come back to initial stages in case they had not turned out as expected

Understanding and keeping all of the above issues in mind, mainly relating to the vendor-based SDLC, the cloud-based service would typically offer the following:

- Requirements Engineering as a cloud service (REaaS) menas it supports reuse of requirements, traceability, and commonality and variability analysis for a familay of related systems. At this pahse of any project management activities

using REaaS can also support contractual regulations, project initiion, acceptance testing and planning.
- Product design based on the collected requirements and documented artifacts.
- Implementation through cloud service based on customer chosen environment.
- Un-optimized and deviation from the required product needs to be checked, and hence testing is performed.
- Option of upgrading and updating the services to incorporate changing and dynamic requirements of the client.

Similar work done previously in this area can be seen in paper [7] where authors have focused on a cloud-based management of projects through the means of software as a service and its various augmentable utilities. They have proposed a model on the cloud which provides SDLC phases as coordinated services. In this proposed model, services will be able to interact with one another and either providers or consumers of data and behavior, instead of letting the client collect all the data and put it altogether after gathering from numerous vendors. A technique and approach for the implementation of the model is also stated in detail. The internal working of the model makes use of two services – IaaS and SaaS. SaaS (software as a service) borrows services and resources from IaaS (Infrastructure as a Service) providers and in turn leases those services to the users. This maximizes resource utilization and also results in increasing customer satisfaction level (CSL). The SaaS contains two vital layers, namely, platform layer and application layer. While the platform layer would be responsible for the admission control depending on how many projects are already admitted, scheduling process, etc., the application layer is required to assemble the service from IaaS and integrate the resources with it to perform the job which conventionally is done by a third party system. The model proposed by them can be implemented not only for small term developer level projects but also higher level project management for which the number of resources to be utilized is a long-term and non-ephemeral function of usage and maintenance.

5.4 Classical vs. Cloud-Based Software Project Management

As management is said to both science and the art, so is project management. The careful process of bringing together economics, software technology, and human relations for a software project is not a simple task. A software project is an extremely people-intensive exertion that traverses an exceptionally long period, with crucial ramifications on the work and execution of a wide range of classes of individuals.

A software project can be regarded as the assembling of tasks that create an identifiable and valuable outcome. In its fundamental form, project management [8–9] consists of planning, executing, and monitoring these activities (Fig. 5.4). Notwithstanding, the high expenses and failure rates of software projects keep on

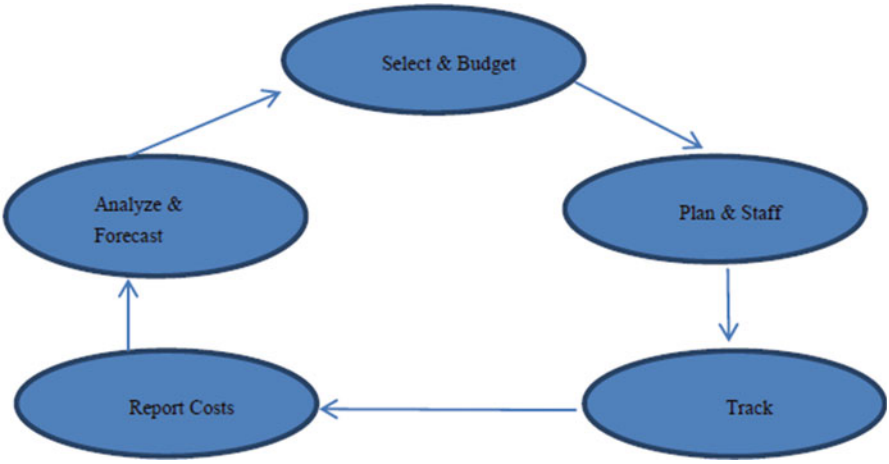


Fig. 5.4 Project management lifecycle

engaging analysts and specialists, and regardless of a few advances, the successful administration of the project is still a challenging process. Dealing with the one of a kind and complex procedures that constitutes a task includes the execution of particular administration exercises. In programming improvement, as in most different organizations, there has been an inclination toward institutionalizing these exercises by method for formalized, nonexclusive project management methodologies like PRINCE2 [8], which was created and championed by the UK government. In spite of the fact that there is a worldwide origination of the project management marvel, there is no brought together hypothesis of project management or very much characterized measure of project success. It is beyond the capabilities of project teams of large software projects to decide the technological, environmental, and organizational states which might have an influence on the outcome of the desired product. Another challenge faced is that the information required to extrapolate most software problems depends upon the individual's idea for solving them. The sort of issues that software projects manage have a tendency to be exceptional and hard to define, and arrangements have a tendency to advance constantly as designers pick up a more prominent appreciation of what must be settled. Adding to the many-sided quality of the issue and its answer is the quick changing and very questionable environment, for instance, market turbulence and changes in client prerequisites and project goals [10]. It is vital along these lines to acknowledge that our suspicions and forecasts about future occasions will, by nature, be indeterminate. While overseeing software projects, we should be to a great degree careful of extrapolating past patterns or depending too vigorously on past experience. The more noteworthy the instability inborn in a project, the more the project needs to move from customary methodologies that depend on an altered succession of exercises to methodologies that permit to reclassify the exercises – or even the structure of the project arrangement – in mid course. Hence, as the

uncertainty and complexity of a project increases, managers need take on roles toward flexibility and learning rather than the traditional risk management.

Some of the important project management software features to be considered are [20]:

1. *Task Management*: To simplify managing and achieving goals, they are broken down into a set of tasks. Tasks are created and managed during the entire process. Tasks such as creating tasks, managing subtasks from larger tasks, set tasks to recur or repeat should be handled by the software.
2. *Team Collaboration*: This forms one of the most important features especially in a distributed team environment. A virtual space needs to be created for discussions among team members. It should allow creation and sharing of documents as well as sending messages to one or more people.
3. *Email Integration*: Integration of the project management software with email turns out to be very beneficial as well as powerful. It can be used for sending updates, information about new tasks, and status reports to a predefined list of members.
4. *File Management*: The online application can provide storage space to manage the files and documents easily with or without the help of a third party. Features like adding notes to files, uploading files, having a version control, and organizing files can also be provided.
5. *Scheduling*: This feature of the software deals with setting time lines and creating milestones for completion of various tasks and also identifying dependencies between resources. This might not be very important for a small team or simple project.
6. *Project Management*: Project management is very crucial for larger organizations where templates need to be created, issues need to be managed, and prioritization among projects is required [9].
7. *Time Management*: Project management software can help in providing a certain degree of control in accepting submitted reports, timesheets, etc. This is valuable to project teams handling many resources and running for longer duration of time.

Software projects have many properties and attributes which make them different from any other engineering project. For instance, the product is intangible due to which we can say that a product is 90% complete even though there are not any visible outcome. Due to such issue, it is very important to have proper project management to ensure a quality product to the clients. The core activities involved in software project management are project planning, project scheduling, risk management, control, and managing people. An efficient software project management focuses on people, problem, and the process. People must be organized into teams and motivated to do quality work and should coordinate well to achieve effective communication and results. The problem must be communicated clearly from customer to developer which must then be decomposed into goals and assigned to the respective teams. Finally, the process

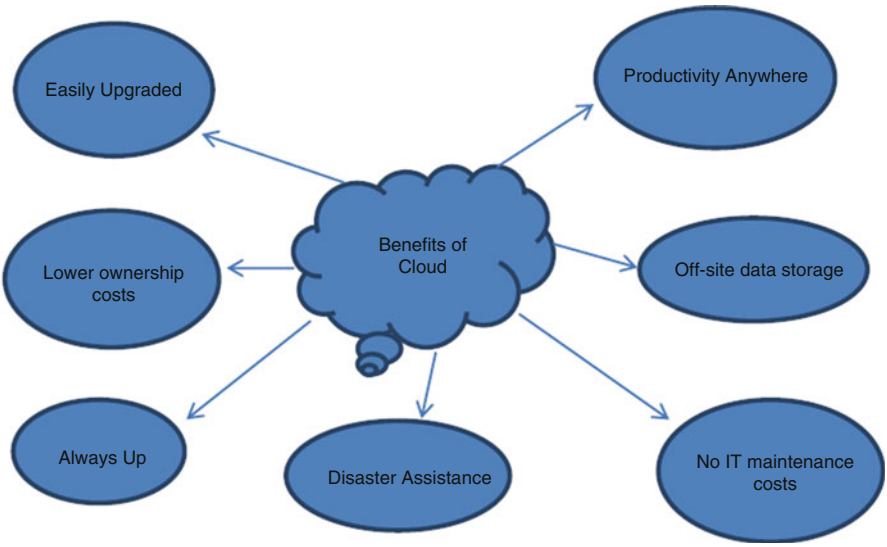


Fig. 5.5 Advantages of using cloud-based PM tools

should consist of a set of work tasks chosen which must be adapted to the people and problem.

With all of the above in mind, we can see that the use of cloud computing [18, 19] in software project management will prove to be immensely beneficial. Cloud-based project management tools can be used to set priorities and align teams to work faster and smarter across the organization. Business is moving faster, becoming increasingly collaborative, and embracing more remote workers every day. Hence, a system is needed which allows us to plan and adjust in real time. With the help of the cloud, it is possible to have a central tool to manage the entire software development process and track progress of the project and also monitor whether the employees are working toward the goals of the project and company. Another prime reason for shifting to cloud-based tools would be cost-effectiveness. Some advantages of using a cloud-based tool are shown in Fig. 5.5.

As it can be seen, the major advantages of using cloud-based project management tools include lower maintenance cost, and also it can be easily upgraded. Maintenance of servers and systems is one area where organizations tend to spend a lot of money. This cost can be drastically reduced with the use of the cloud. The other advantage cloud provides is that it can be easily upgraded and no extra hardware and systems need to be set up in case there is a need for scaling the services. The next section gives a critical evaluation of some of the popularly used cloud-based project management tools which have been compared using certain criteria.

5.5 Evaluation of Cloud-Based Software Project Management Tools

In the manner in which programs can be written either in editors such as notepad through Vi to eclipse, online project management tools range from shared to-do lists to multimedia collaborative environments. In order for a portal to be considered as a software project management tool, it must have a few specific features. Some of the criteria for evaluating the tools are bug tracker, to-do lists or some kind of task management support, and a document repository which helps the stakeholders to share and modify content and understand what work have been done so far. Another important feature is conversational tools like emails, chat, wikis, blogs, etc. which provide a mechanism for stakeholders to communicate and collaborate. All the other components like calendar sharing, report generation, tagging mechanism, and time tracking tools which may be provided are offshoots of the core features, namely, task management, document repository, and conversational tools [20].

There are many existing project management tools available online/in the cloud. Some of the popular tools are listed with the features they offer and the different industries they are used in:

1. *Freshdesk* [21]: It is the most recent in cloud-based support tech that comes with everything needed to manage and track projects. They follow a simple goal of making the process of brands talking to their customers and also making it easier for customers to get in touch with their businesses. An array of features like issue tracking, SLA management, smart automations, SEO ready FAQ section, knowledge base, and customizable self-service portals are provided by Freshdesk which helps increase agent productivity and reduce burnout.

Used by: Real estate, professional service providers, healthcare, and insurance

2. *Zoho Projects* [22]: Zoho projects are the project management software from Zoho. It provides features like project planning, assigning tasks, effective communication, update reminders, and detailed reports on progress. Unlimited users can be added to all plans with no extra cost.

Used by: Small and large teams across various industries

3. *TouchBase* [23]: TouchBase is totally a state-of-the-art, web-based project management software. It offers an incorporated bundle with management, asset tracking, purchasing, contract management, self-service portal, and knowledge base at a reasonable cost point. TouchBase gives all that you need an undeniable IT Help Desk and a beneficial Help Desk staff for your project management team. TouchBase can be easily customized as per your industry requirements.

Used by: Global corporations around the world

4. *SpiraPlan* [24]: SpiraPlan provides a complete agile project management system in one package that manages your project's requirements, releases, iterations, tasks, and issues in one environment, fully synchronized. Designed specifically to support agile methodologies such as Extreme Programming (XP),Kanban, Scrum, DSDM, and Agile Unified Process (AUP), it allows teams to manage all their information in one environment.

Used by: Project managers and IT professionals

5. *Easy Redmine* [25]: Easy Redmine is an open-source software for complex project management with extensions for resources, finance, and customer management. In the cloud or on your own server, all comes with professional implementation and support. Easy Redmine supports whole project lifecycle, so you can start with an area where you feel the most urgent need. Afterward, Easy Redmine can grow with you, thanks to the features which work as separately installable extensions. Over 20,000 users worldwide

Used by: Software developers, education, healthcare, media, government

6. *eXo* [26]: eXo platform is an open-source social-collaboration software designed for enterprises. It is full featured, based on standards, and extensible and has an amazing design. eXo helps companies connect their employees, customers, and developers through social, collaborative, and content-driven intranets, websites, and dashboards.

Used by: Large enterprises, mid-size businesses, public administrators

7. *Basecamp* [27]: Web-based software that makes it simple to communicate and collaborate on projects. It is used by millions of people, and 98% of its customers recommend it, primarily for its simplicity. It supports multiple languages and can be accessed on your mobile phone.

Used by: Freelancers, entrepreneurs, small businesses

8. *Genius Project* [28]: Genius inside offers its prime solution Genius Project since 2008 as cloud-based as well as on-premise solution for its project management software. Apart from the typical project management features, some of its noteworthy features include simulator which gives visual representation of the what-if scenario's in the project and phase and gate review support process for new product development, and it also provides Agile and SCRUM support. Genius Project is available in three deployment options: hosted on premise, SaaS, or installed on IBM's Lotus Notes. Extremely user friendly and customizable user interface with built in social collaboration platform, Genius Project fits the needs of every industry and provides benefits for everyone in the organization: PMO, executive, project manager, and team member.

Used by: Project centric companies of all industries

9. *Trello* [29]: Trello lets you organize anything with anyone. It is a flexible project management solution that fits into your workflow in a visual, collaboratively

focused way. Trello replaces post-it notes in a digital whiteboard format that can be used for anything from redesigning a website, to posting company updates regularly for management, to complex projects with many participants.

Used by: Project management teams of any size

10. *Kanzen* [30]: Kanzen is a project management and collaboration tool that focuses on but is not limited to Kanban method to improve business processes. A unique set of features allows user to view their workload in three views – a Kanban board, task list, personal task list, and a calendar. Intuitive interface and the ease of use will allow you to concentrate on your tasks rather than struggling with the software. Features include e-mail notifications, analytics, access rights, and more.

Used by: Businesses, project management teams, and individuals

11. *Salesforce* [31–32]: Uses the world's best CRM for small businesses in combination with a top project management tool from our AppExchange to better manage and gain visibility into all stages of your company's projects. The power of Salesforce plus a project management partner on our AppExchange will allow your company to reach its peak efficiency and productivity. Salesforce's Sales, Service, and AppExchange applications help companies connect with customers, partners, and employees in entirely new ways.

Used by: Companies of various industries and any team size

As seen in Table 5.1, the tools were compared and evaluated on the abovementioned criteria. Upon the evaluation of these tools, we find that most of the portals offer some kind of repository and ticketing mechanism and support for communication tools. A role-based access control was another feature common in many of the portals. Some major differences can be seen in the target markets of these tools apart from the usual pricing and licensing differences. For instance, BaseCamp [27] primarily targets small organizations that are staffed by nonprogrammers working on short- or medium-length projects. Though there are many software developers using it, it still forms a minority of the users. This might be the reason it offers a simple and easy-to-use file upload system rather than a version-control system. Another trend observed is that companies are giving more emphasis on agile methods which also have a big influence on the features offered by them. This explains the shift from asynchronous communication like bulletin boards to synchronous communication like chats.

After studying these tools, we understand the real importance of requirements elicitation and the importance of a structured development process in the success of a software project. This is one of the critical areas where we would like to target in SPMaaS. In addition, other research issues include are whether the information provided in the portal is sufficient enough to carry out the process or do the team members need to depend on other means of communication like personal communication which might not be recorded in the portal. With such problems arising, we feel that by addressing such issues, the next generation of tools can be designed to

Table 5.1 Table comparing tools based on features provided

	Tools						
Features	Zoho Projects	TouchBase	SpiraPlan	Easy Redmine	eXo Platform	Basecamp	Trello
Budget maintenance	✓	✓	✓	✓			
Collaboration	✓	✓	✓	✓	✓	✓	✓
Prioritization				✓			✓
Email integration	✓		✓		✓	✓	
File sharing	✓	✓	✓	✓	✓	✓	
Gantt charts	✓	✓	✓	✓			
Issue management	✓	✓	✓	✓			
Milestone tracking	✓	✓	✓	✓			
Project planning	✓	✓	✓	✓	✓		
Requirements management		✓	✓	✓		✓	
Status tracking	✓	✓		✓			
Time and expense tracking		✓		✓			

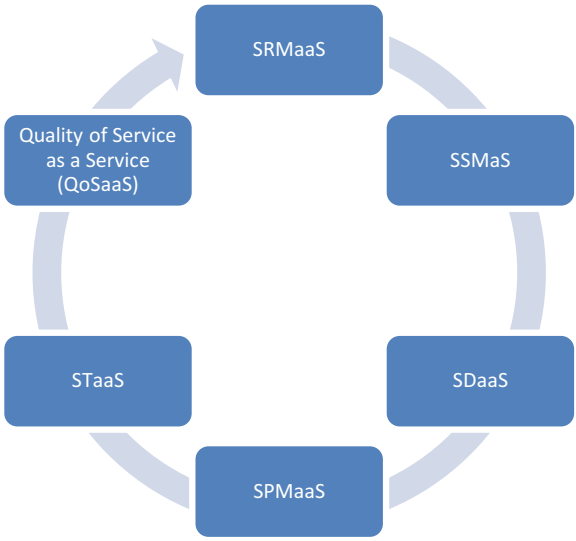


Fig. 5.6 Integrated software engineering as a service

better satisfy the real users' needs, and it is our aim to do this with the help of a cloud-based service SPMaaS.

5.5.1 Integrated Software Engineering as a Service

After evaluating the existing tools, we have realized the need of integrating project management in the software development cycle. The project management phase will begin immediately once the project development has begun. This can be seen clearly in Fig. 5.6.

Figure 5.6 clearly shows the overall proposed infrastructure of the proposed project management as a cloud service and how it will sit in the cloud. It will begin with software requirements as a (SRMaaS) service followed by software security management (SSMaaS). After this, the development will begin (SDMaaS), and project management will begin soon after (SPMaaS). The lifecycle will end with software testing (STaaS) and ensuring quality of the product (QoSaaS). The next section gives a detailed explanation on our proposed approach of SPMaaS and highlights its core activities.

5.6 Integrated Service Development Process and Software Project Management for SPMaaS

Project management can help companies, managers, and project teams to consummate client requirements, budget, manage time, and scope constraints. It is very important for the companies to choose the right tools so that it can help them save project cost and project time. Basically, there are two types of project management software:

- On premise [33]: These software systems reside in the data center owned by the company and runs on their own server. It is maintained by the IT employees of that company: Microsoft
- Cloud based [34]: This uses cloud technology and is offered by service providers as *SaaS*(software as a service). Many small- and medium-sized enterprises use cloud-based project management tools across different industries.

Figures 5.7 and 5.8 show the core activities which will be offered as a cloud service SPMaaS. It includes software project planning (SPPaaS), software cost estimation (SPCEaaS), software team management with support to handle virtual teams and multi-tenancy (STMaaS), and continuous delivery (CDaaS).

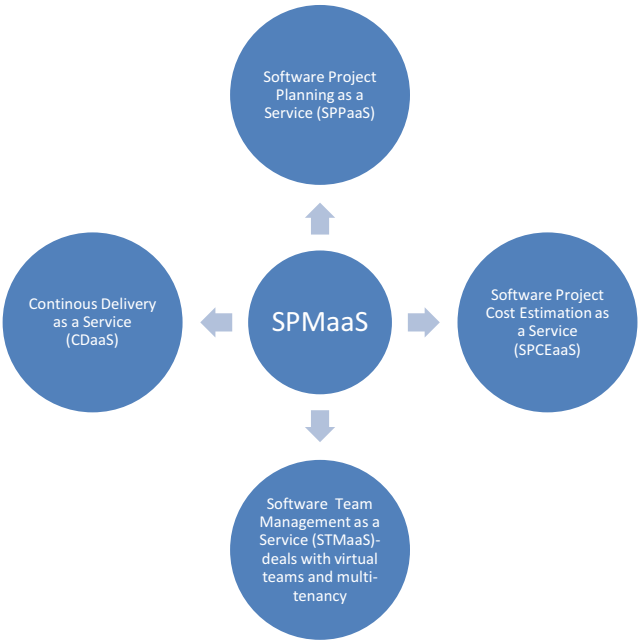


Fig. 5.7 Core activities of SPMaaS

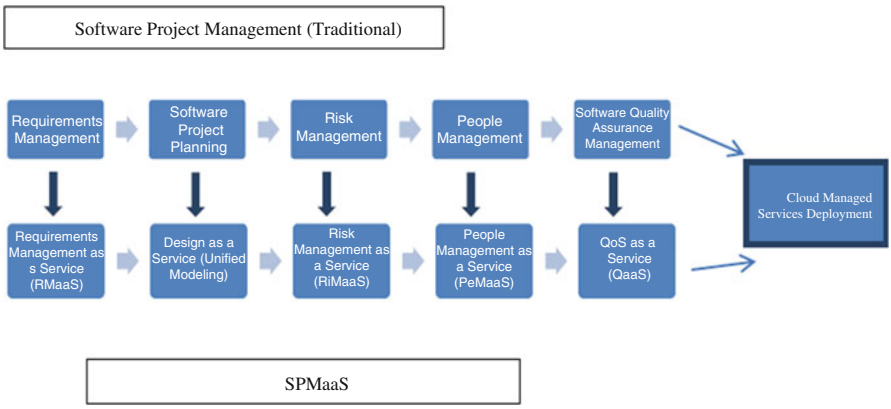


Fig. 5.8 Software project management as a service (SPMaaS)

A detailed explanation of the features provided by SPMaaS is as follows:

- Requirements Management as a Service: Requirements management refers to the process of documenting, tracing, analyzing, agreeing, and prioritizing on requirements and then controlling change and then communicating it to the appropriate stakeholders. Any capability to which a product or service should conform is called a requirement. Poor requirements management is one of the major causes of project failure, and hence, it is a very important phase. This can help us exceed stakeholder expectations, improve performance, and meet the expected project goals. Refer to Fig. 5.9.
- Design as a Service: Unified modeling languages (UML) are used to provide a standardized way to visualize the design of a system. A set of diagrams can be drawn to visualize the system such as activities, individual components of the system, interaction among software components, external user interface, etc. The types of diagrams include structure diagrams like class diagram, component diagram, object diagram, and behavior diagrams like activity diagram, use-case diagram, interaction diagram, etc. UML diagrams help in simplifying the software development process and reducing development time.
- Risk Management as a Service: It refers to the identification, assessment, and prioritization of risks. The objective of risk management is to assure that uncertainty does not deflect the endeavor from the business goals. Risks need to be identified as early as possible to avoid any obstacles in smooth development process. Since there are infinite number of events that can have negative effect on a project, no project can ever be risk-free. Good an efficient risk management increases the likelihood of a successful project.
- People Management as a Service: People management aims at getting things done from people through effective management to produce outstanding results. It deals with understanding, managing, and delivering people's expectations. People management is one of the hardest aspect of a project management. A good team manager needs to understand strengths and wekaness of the team

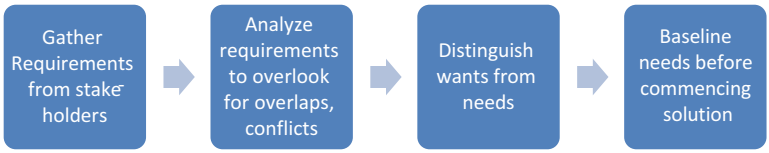


Fig. 5.9 Requirements management process

members and being able to be a motivator and should be willing to take the leadership of the team work. However, if we have a good team processe in place, it is then possible to achieve the common objectives of the team as well as the project.

- Quality as a Service: Apart from delivering the product on time, the quality of the product also plays an important role. There is a need of a process which ensures that the developed software meets and complies with standardized quality specifications. It needs to be an ongoing process within the lifecycle that routinely checks the software and hence ensures the development of a high-quality product.

With all of the above services provided as a cloud service, the process of project management is simplified, and the project can be managed very effectively and efficiently. The next section gives a detailed explanation of the architectural design with service-oriented architecture and SoaML diagram.

5.7 Architectural Design of SPMaaS with SOA

The idea of a service has been defined by many people in numerous ways. While it has been described as an encapsulated unit of functionalities, it has also been considered as a logical manifestation of some physical resources grouped as a process that an organization exposes to a network. A service has been defined as an externally observable behavior of a software/hardware component in which the internal working and processing details are well hidden and made available through a set of well-defined interfaces [35]. A service in the context of web services can also be viewed as an application or business logic that exposes its functional capabilities to clients by running on a server. Refer to Fig. 5.10. One thing which is common and is being tried to be explained in all of the definitions is that service can be viewed as conceptual identity which supports certain actions in response to a set of requests received. These requests can be in the form of messages or some kinds of programs written to trigger the internal processing at the service providers' end.

Software applications can be implemented using abstraction as the fundamental design entity. Each service clearly encapsulates certain features while at the same time hiding the underlying implementation details from the user/client. This concept greatly benefits while building systems to implement higher level services. The set of services which need to be provided are decided in the software development

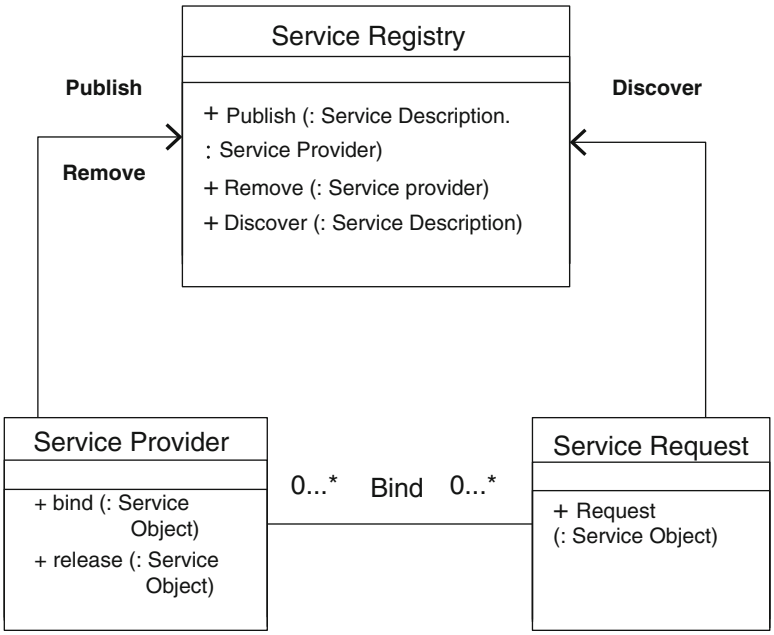


Fig. 5.10 Conceptual model of service-oriented paradigm

phase. The complete system can then be built by having these services as the fundamental design entities.

In this architectural style, an interaction model between parties is defined, namely, service provider, service consumer, and service request. The provider publishes the service description and provides implementation for a service. The consumer can either use the URI for the service description directly or can find the service description in a service registry and invoke and bind a service.

5.7.1 Types of Services Offered by SPMaaS

There can be many services provided by providers in software project management, which depending on the complexity may require different levels of processing. Services can be composed into three types: elementary services, collaborative services, and composable services.

For instance, the features provided in software project management can be also broadly classified into services for planning and scheduling, services for collaboration, services for documentation, etc.

5.7.1.1 Elementary Services in SPMaaS

These refer to services which do not require complex processing and which are independent in nature. There are no additional requirements or constraints that need to be fulfilled in order to add these kinds of services. The client can add an elementary service by making a simple request in any form, and the respective service will be made available to the user. For example, services like software testing (STaaS) can be invoked separately without any previous requirements.

5.7.1.2 Composable Services in SPMaaS

These are the services which are not readily available but can be provided by invoking a set of services belonging to the same category of services. Consider a scenario in which the client needs to add a new service. In order to provide the client with this service, it might be possible that there must be some existing services the client should already be using so that the new service can be provided with the help of them. The new service which the client might want could need the support of another service for its fulfillment or it might be an extension to some older service. For example, if a client wish to add a calender shring feature, then, this should be part of a Software Planning as a Service (SPPaaS).

5.7.1.3 Collaborative Services in SPMaaS

These services can neither be composed using the services at a service window nor can be available readily. Basically, these are the services which can be provided only if a set of conditions are followed. For example of a composite service (consists of invoking a sequence of a number of other services), taking a hypothetical situation, if you invoke a new software project cost estimation as a service (SCEaaS) of the newly created instance of a SPMaaS project, then this service will create autonomically new instance of software requirements engineering as a service (SREaaS) and software security requirements engineering as a service (SSREaaS) which in turn will also invoke software project planning as a service (SPPaaS). Hence in this sequesnce of service invokation, one services is interdependent on a set of other services and establishing service choreography.

5.7.2 Design of Cloud SPMaaS with SoaML

The SOA design for SPMaaS can be clearly explained with the help of the diagram shown in Fig. 5.11. The diagram has been drawn using service-oriented architecture modeling language (SoaML) which is an extension of UML 2.0 to support service

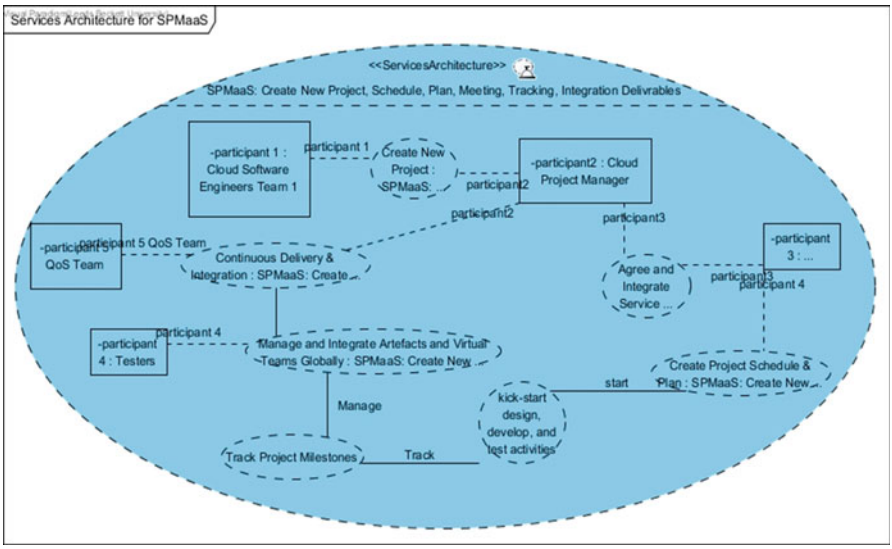


Fig. 5.11 SOA design for SPMaaS with SoaML

concepts. SoaML provides a standard way to architect and model SOA solutions using the unified modeling language (UML). A services architecture (SOA) is a network of participant roles providing and consuming services to fulfill a purpose. The services architecture defines the requirements for the types of participants and services that fulfill those roles.

The diagram, in Fig. 5.11, shows various participants (rectangle boxes) such as cloud software engineering team, project manager, testers, QoS teams, etc. Participants can provide as well as use services (represented with oval shape). Some of the services provided which can be seen in the diagram are creating a new project, creating a project schedule and plan, and tracking project milestones and continuous delivery and integration. The SoaML diagram also clearly shows which participants can create a service, use a service, or invoke a service with the help of messages. For example, we can see that a new project can only be created by participant1, namely, cloud software engineer team and participant2, the cloud project manager. Thus, with this diagram we can understand:

- The roles each participant plays in a service
- The message types that go between participants when a service is enacted
- Interfaces provided and used by each participant for the service
- Choreography of interactions between the participants while enacting the services

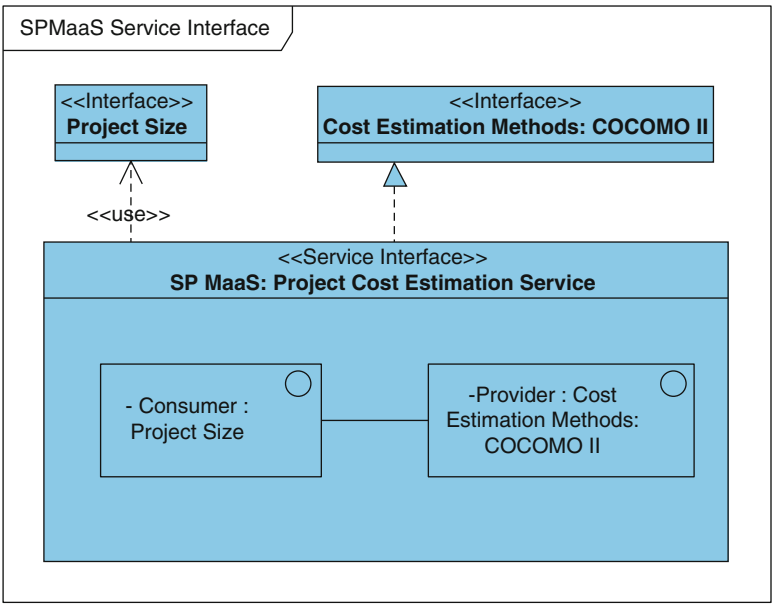


Fig. 5.12 SPMaaS service interface model

5.7.2.1 Part I – SPMaaS Service Interface Model

Service interface diagram is one of the most important SoaML diagram types. The idea of a service interface diagram is based on the core aspect being a service. A service in this case can be defined as a value delivered to another through a well-defined interface. In SoaML, a service can be specified using three approaches, namely, simple interface, service interface, and a service contract. It can be seen in Fig. 5.12 that a service interface *SPMaaS: Project Cost Estimation Service* is created. A service interface involves communication and interaction between a consumer and provider of services. In Fig. 5.12, it can be seen that the consumer is *Project Size* which is provided with *Cost Estimation Methods*. There are also two simple interfaces- Project Cost and *Cost Estimation Methods* which have beeen provided. The service interface of the project cost estimation service specifies its required needs through usage dependencies to the *Project Size* interface and the receptions and operation it receives through the *Cost Estimation Methods* interface.

5.7.2.2 Part II – Specifying SPMaaS Choreography Using UML Sequence Model

Service choreography defines the interaction between the provider and consumer in completing a service. We can specify how the consumer interacts with the provider

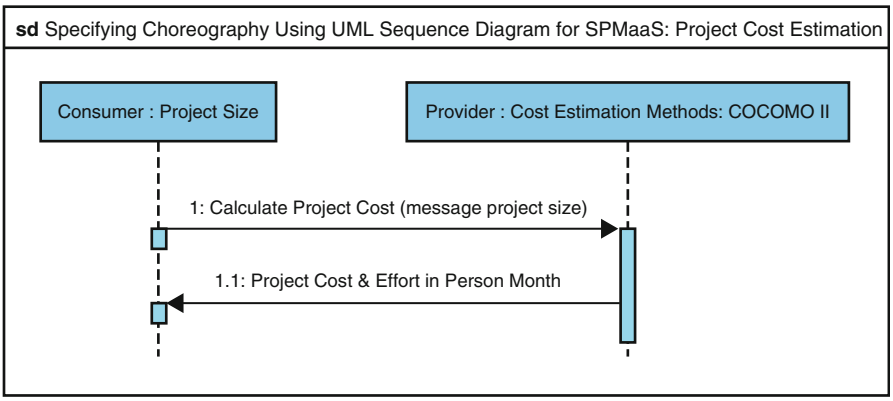


Fig. 5.13 SPMaaS choreography using UML sequence diagram

of service with the help of sequence messages between the two lifelines. In Fig. 5.13, it can be seen that the consumer begins by invoking the provider to calculate the project cost. The provider in turn reacts by replying with the person cost and effort in person month.

5.7.2.3 Part III – SPMaaS Service Participant Model

In SoaML, participant refers to a certain party or component that provides and/or consumes a service. Participants can be software components, organizations, systems, or individuals. In Fig. 5.14, globally distributed teams/ virtual teams or client has been taken as the participant. We can also see the services provided and used by the participant. A square which represents the port can be seen providing the interface for creating a new project and requiring the client requirements.

5.7.2.4 Part IV – SPMaaS Service Contract Design

A service contract specifies and defines the agreement between parties about how a service is to be provided and consumed. Interfaces, choreographies, terms, and conditions are used to define the agreement. The interacting participants must compulsorily agree and adhere to these agreements in order for the service to be enacted. Figure 5.15 shows the service contract for a new project contract service in SPMaaS in which the consumer and provider need to agree to the project contract agreement.

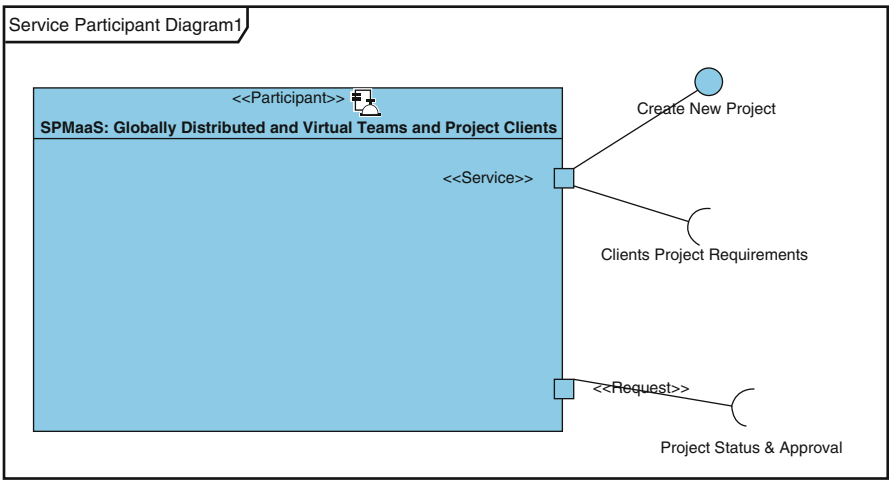


Fig. 5.14 SPMaaS service participant model

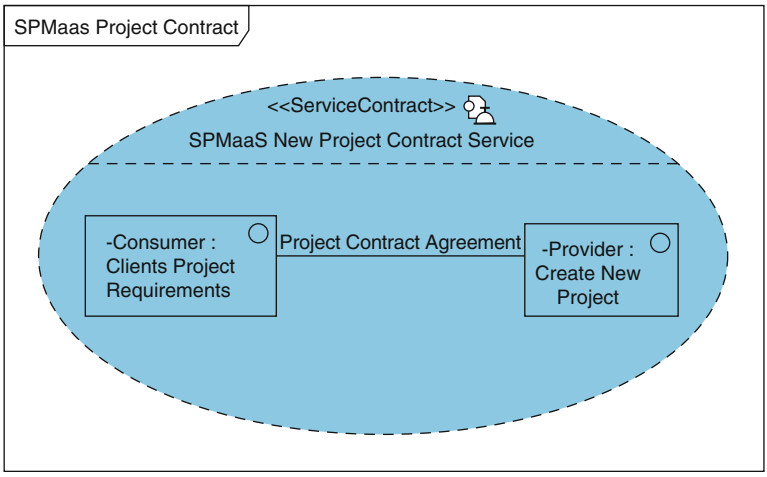


Fig. 5.15 SPMaaS service contract design

5.7.3 Results and Analysis of SPMaaS Design

The design for cloud services is challenging, and it is in its infancy for a proper and systematic approach to engineering cloud service design and development as we have shown in this chapter the importance of engineering cloud services with the current state-of-the-art tools and standards such as SoaML which has been specifically developed for cloud service engineering. Figure 5.16 shows how we have measured a number of service components for each of the SoaML stages.

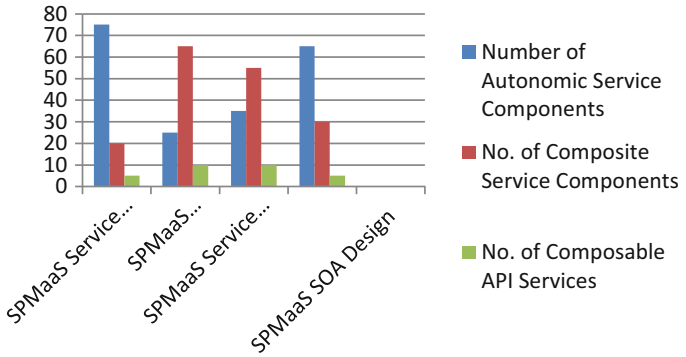


Fig. 5.16 Graph showing count of service components for different SoaML design categories

The graph shown in Fig. 5.16 shows a count of the service components for each of the different five SoaML design categories like SPMaaS service interface model, SPMaaS choreography, SPMaaS service participant model, SPMaaS service contract, and SPMaaS SOA design. Three different service components have been considered:

- *Autonomic service components*: These are the components that can work with any dependency to complete a full service.
- *Composite service components*: These depend on other services to complete a full service and cannot exist independently.
- *Composable API services*: These are the API services which can be created with the support of existing APIs available and by integrating them with some new features and providing them to the customers for easier and more specific use.

5.8 Conclusion

Cloud computing is emerging rapidly with increasing demand for service-oriented computing and associated technologies. This is the right time to explore what works better and what doesn't work for cloud environment. Therefore, the proposed model helps to understand how it should be developed to avoid classical issues related to software development projects. We believe the proposed model will help us to develop cloud applications systematically. Another major contribution of this chapter is to use SoaML to design cloud-based software project management as a service system (SPMaaS). SoaML provides a standard way to architect and model SOA solutions using the unified modeling language (UML). A services architecture (SOA) is a network of participant roles providing and consuming services to fulfill a purpose. The services architecture defines the requirements for the types of participants and services that fulfill those roles. This study discovered overall 70%

improvement of the cloud-based services by designing with SoaML by counting number of service components during the design phase of this research.

References

1. Cloud: Amazon Elastic Compute (2011) Amazon web services. Retrieved 9 Nov 2011
2. Buxmann P, Thomas H, Sonja L (2008) Software as a service. Wirtschaftsinformatik 50 (6):500–503
3. Zhang QI, Cheng L, Boutaba R (2010) Cloud computing: state-of-the-art and research challenges. J Int Serv Appl 1(1):7–18
4. Wang L, Laszewski VG (2008) Scientific cloud computing: early definition and experience. http://cyberaide.googlecode.com/svn/trunk/papers/08-cloud/vonLaszewski-08-cloud.pdf
5. Ramachandran M (2008) Software components: guidelines and applications. Nova Publishers, New York
6. Bichier M, Lin K-J (2006) Service-oriented computing. Computer 39(3):99–101
7. Khan A et al (2012) Cloud service for comprehensive Project Management Software. Application of Information and Communication Technologies (AICT), 2012 6th international conference on IEEE
8. Bentley C (2010) Prince2: a practical handbook. Routledge
9. Thayer RH, Yourdon E. (1997) Software engineering project management. In: Software engineering project management, pp 72–104
10. Helbig J (2007) Creating business value through flexible IT architecture, Special Issue on Service-oriented Computing. IEEE Computer 40(11)
11. IaaS (2010) Cloud computing world forum. http://www.cloudwf.com/iaas.html
12. IThound Video whitepaper (2010) http://images.vnunet.com/video_WP/V4.htm. Accessed Feb 2010
13. SaaS (2009) SaaS. http://www.saas.co.uk/
14. Science Group, 2020 Science Group: toward 2020 science, tech.report, Microsoft, 2006. http://research.microsoft.com/towards2020science/downloads/T2020S_Report.pdf
15. Vouk MA (2008) Cloud computing – issues, research and implementations. J Comput Inf Technol, CIT 16
16. Wilson C, Josephson A (2007) Microsoft Office as a platform for software + services. Archit J 13. www.architecturejournal.net
17. Zhang L-J, Zhou Q (2009) CCOA: cloud computing open architecture. In: IEEE international conference on web services
18. Armbrust M, Fox A, Grifth R, Joseph AD, Katz R et al (2009) Above the clouds: a Berkeley view of cloud computing. Technical report, University of California at Berkeley. URL http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html
19. Foster IT, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared, CoRR abs/0901.0131
20. Project Management Tools (2016) http://modeling-languages.com/survey-web-based-software-project-management-tools/. Accessed Sept 2016
21. Freshdesk (2016) https://freshdesk.com/. Accessed Sept 2016
22. Zoho projects (2016) https://www.zoho.com/projects/. Accessed Sept 2016
23. TouchBase (2016) http://www.productdossier.com/. Accessed Sept 2016
24. SpiraPlan (2016) https://www.inflectra.com/SpiraPlan/. Accessed Sept 2016
25. Easy Redmine (2016) https://www.easyredmine.com/. Accessed Sept 2016
26. eXo Platform (2016) https://www.exoplatform.com/. Accessed Sept 2016
27. BaseCamp (2016) https://basecamp.com/. Accessed Sept 2016
28. Genius Project (2016) http://www.geniusproject.com/. Accessed Sept 2016
29. Trello (2016) https://trello.com/. Accessed Sept 2016

30. Kanzen (2016) https://mykanzen.com/. Accessed Sept 2016
31. Salesforce (2016) http://www.salesforce.com/in/. Accessed Sept 2016
32. Cost Estimation Techniques (2016) http://www.computing.dcu.ie/~renaat/ca421/report.html#2.6. Accessed Sept 2016
33. Turner JR (1993) The handbook of project-based management: improving the processes for achieving strategic objectives. McGraw-Hill, London
34. Sotomayor B et al (2009) Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput 13(5):14–22
35. Wan K-M et al (2006) Service-oriented architecture

Part II
Approaches and Frameworks for Software Development and Software Project Management

Chapter 6
Component-Based Hybrid Reference Architecture for Managing Adaptable Embedded Software Development

Bo Xing

6.1 Introduction

Ambient Assisted Living (AAL) can be defined as one that "will assist elderly individuals for better, healthier and safer life in the preferred living environment and covers concepts, products and services that interlink and improve new technologies and the social environment" [4]. This concept provides a close vision of the ambient intelligence (AmI) technologies where the emphasis is on independent living of the disabled and the elderly, such as smart home [21], healthcare systems [66], and assistive robotics [30, 75]. However, despite research that shows the capability of technology to support a longer and higher-quality living for the elderly and the disabled, there is still a wide array of challenges of making AAL a reality. For example, the identification of the needs of elderly people [35] and of the necessary technological support [22] is related to the AAL. Moreover, the authors of [50] pointed out that a lack of software reference architectures (SRA) of AAL, which could provide better ways to develop AAL systems, has also been identified as main challenge.

In general, the SRA refers to a software architecture equipped with predefined structures and respective elements and relations that facilitate providing the templates for concrete architectures in a particular domain or in a family of software systems [16]. In this chapter, we propose a hybrid reference architecture for AAL, named software architecture for AAL (SAfAAL for short), which is built on the component-based architectural style [62]. Finally, we have prototyped the proposed model and implemented a case study to demonstrate its effectiveness.

B. Xing (✉)
Computational Intelligence, Robotics, and Cybernetics for Leveraging E-future (CIRCLE), Institute of Intelligent System, Faculty of Engineering and the Built Environment, University of Johannesburg, Johannesburg, Gauteng, South Africa
e-mail: bxing2009@gmail.com

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_6

The remainder of this chapter is organized as follows. Subsequent to the introduction in this section, the background of elderly people, key applications, and related technologies in the context of AAL are briefed in Sects. 6.2, 6.3, and 6.4, respectively. Then, the employed methodologies are presented in Sect. 6.5 which is followed by proposed framework detailed in Sect. 6.6. Next, Sect. 6.7 elaborates such framework in an ALL smart home environment to demonstrate the feasibility of our proposed methodology. In addition, the trade-off strategy for managing software adaptability is also discussed in Sect. 6.8. Finally, the highlighted future research directions and concluding remarks provided in Sect. 6.9 close this chapter.

6.2 Elderly People in the Context of AAL

One of the most important characteristic of demography is ageing and shrinking population. It is a global phenomenon, but it is observed that some countries are worse than others. For example, the Federal Statistical Office shows that Germany has one of the world's most rapidly ageing and shrinking population [6]. In addition, Japan's government forecasts that over the next 50 years, Japan's overall population will shrink by a third compared with currently 127 m [5]. As a consequence, the chief problem of labour market is that a fast-shrinking working population will struggle to support a growing proportion of the old. In this context, AAL has arisen as a philosophy that provides considerable support for the everyday life of elderly people [15]. The main purpose of AAL is to enable the elderly to live an independent life in their own homes and communities as long as they possibly can. Research shows that innovative technologies might provide a solution for the ageing population, such as sensor-based networks for activity monitoring, fall and wandering detection, and various portable e-health applications. However, with increasing size and complexity, researchers require a guideline to identify the architectural patterns and principles to identify important quality attributes and to understand and reason the crosscutting concerns.

6.3 Key Applications in the Context of AAL

In recent years, one of the most popular issue is ageing and decreasing population. Under these circumstances, the most prominent study is focused on the assistive technologies (ATs), such as smart sensors, data acquisition systems, ubiquitous data connectivity, and big data analysis. In the following subsections, three key applications, i.e. smart homes, healthcare system, and assistive robotics, are introduced.

6.3.1 Smart Homes

The idea of smart home, which builds upon advances in sensors and actuators enabling, is to achieve activity recognition and to provide assistive services to a resident. In general, the smart home development-related research topics can be classified into five categories, i.e. home entertainment, management platform [63], home security [78], home care [68], and home network [10]. Moreover, several ongoing and prolific research programmes include Georgia Tech's "Aware Home", MIT's "Place Lab", Samsung's "Smart Home Project", European project "ALADIN" and SMILEY, and Microsoft's "MS Home". For more details please refer to other sources of literature [1].

6.3.2 Healthcare Systems

Global ageing and the associated impact on healthcare systems, a broad ecosystem that encompass many aspects of our daily life such as financial system, network system, and insurance systems, have been well documented. For example, in 2011, McKinsey Centre provided a comprehensive report for US health system in terms of the cost of US healthcare [12]. It examined healthcare expenditures in the United States during the 2006–2009 period and found that in 2009, spending on healthcare reached a record high $2.5 trillion, or 17.6 % of US GDP. To change this situation, various efforts to address the anywhere-anytime accessible online healthcare or medial systems have been proposed and evaluated. Central to many efforts has been the use of the Internet of things (IoT) [19, 54], robotic technology [3, 13], mobile phone or Tablets [74], wireless sensor networks [2], and social media tools [14, 71].

6.3.3 Ambient Assisted Living Robot

The main purpose of introducing AAL robot is to assist the disabled and elderly people at home [25, 58, 75]. In general, they can be categorized into three categories: robots assisting with daily living activities (such as feeding and dressing), robots assisting with instrumental activities of daily living (such as housekeeping and preparing food), and robots assisting with enhanced activities of daily living (such as communication and engaging in hobbies) [58]. Nowadays, a number of assistive robots are being deployed. For example, "Care-O-bot" [30–32], a robot developed by Fraunhofer IPA, is able to fetch and carry objects, communicate with older people, and supply emergency support; "RIBA" [48, 59], a robot developed by RIKEN-TRI Collaboration Center, can help patient transfer; "uBot5" [43], another robot from the University of Massachusetts Amherst, is capable of achieving multiple postures for the purpose of assisting elderly in compensating for

impaired upper extremity function; "PerMMA" [18, 73, 77], a research outcome from the Carnegie Mellon University and the University of Pittsburgh, can assist persons with disabilities; "PaPeRo" [29, 60, 61], another case developed by NEC, is used to communicate; "EMIEW" [36, 38, 39] developed by Hitachi, can interact with human beings; and "Hospi-R" [24, 49], an autonomous delivery robot developed by Matsushita, can even perform complex service tasks.

6.4 Related Technologies of AAL

With increasing size and complexity of smart environment, small, inexpensive, and low-powered consumption sensors play a key role in enabling innovative solutions. Sensor applications in the AAL domain range from physiological monitoring to screening applications. For example, [72] proposed an activity recognition system within a smart home by using a shoe-mounted accelerometer. Other examples include daily activity monitoring [37] (e.g. calorie intake and energy expenditure), detection of situations of helplessness [68] (e.g. fall detection), and remotely tracking vital signs [11] (e.g. heart rate). In addition, different types of sensor platforms are also important, such as Arduino, Shimmer, and smartphones and tablets [46]. These devices can be connected using wired (e.g. USB) or wireless (e.g. Bluetooth) interfaces and through pervasive sensor networks make data exchange. Moreover, cloud computing has become one of the most active areas in information technology. For example, the authors of [27] proposed a cloud-based integration model to accommodate dynamic loads and sharing of sensor resources by different users and applications for lifestyle monitoring, Therefore, it is worth highlighting that cloud computing is playing a major role in terms of achieving the goals of AAL.

6.5 Research Methodologies

In order to develop our proposed hybrid framework, we have intensively used the following two techniques, namely, component-based software engineering (CBSE) and SRA. This section outlines their corresponding essentials.

6.5.1 Component-Based Software Engineering

Component-based software engineering (CBSE) [62] describes a methodology to software systems design and development. The spirit of CBSE lies in that it decomposes the whole system design into individual functional or logical components which are higher-level abstractions and are defined by their interfaces. Since

all implementation details within components are hidden from each other, CBSE can be regarded as a union of component-related operations, e.g. defining, implementing, and integrating (or composing) independent (or loosely coupled) components into systems. As software systems are getting bigger and more complex, CBSE has become an important way of developing software. More dependable software systems which can be delivered faster and deployed quicker are in great demand from the end users. The only way that one can address this issue is to introduce software component concept for the purpose of reusing rather than reimplementation.

6.5.2 Software Reference Architecture (SRA)

Generally speaking an SRA forms a conceptual frame which includes a set of elements, forms, principles, and rationale design decisions. It is an effective and powerful instrument to get an overview and deeper understanding of the software system, to give orientation for aggregating knowledge of a specific domain and reasoning their relationships, and, finally, to improve the modularity, the reusability, and the extensibility for each architectural aspect. Nowadays, SRA for different types of systems can be found, such as embedded systems [23], smart environment [26], and big data [55]. Regarding AAL, some alternative system architectures include Alhambra [20], Hydra [40], OASIS [51], openAAL [52], PERSONA [56], and universAAL [70]. However, as pointed out in [50], all these platforms do not address all quality attributes (e.g. reliability, security, maintainability, efficiency, and safety) and important characteristics (e.g. presence of single point of failure and safety pattern usage). That means the AAL platforms and related SRA are not mature.

6.6 Proposed Framework: Component-Based Hybrid SRA

Nowadays, the producers of mass-manufactured embedded ALL devices often tend to have varied views towards the integral software element, ranging from treating it as a difficult but unavoidable necessary part of a finished product to believing it as a crucial differentiator for market success. In many practical embedded product domains (e.g. automobile, smartphone, and ALL), one is often likely to find an interwoven supplier and subcontractor bonds existing on many levels. Bearing this in mind, the proposed software architecture for AAL (i.e. SAfAAL) is thus the combination of the following three application architecture, namely, mobile, rich client, and Web. They will normally be structured as a multilayered application. Therefore, a short description of the functionality of layered structure is also provided in this section.

6.6.1 Three Key Architectures

This section discusses three basic types of architectures that form the proposed hybrid architecture.

6.6.1.1 Mobile Application Architecture

In this type of architecture, locally cached data are often used to bolster offline or disconnected operations, and a synchronization process is performed when connected. Other types of services exposed by different applications may also be used by this application. Mobile applications are often used in the following context, e.g. handheld devices are often involved, display screen size is often an important consideration, and offline or partially connected scenarios are included.

6.6.1.2 Rich Client Application Architecture

Applications of this type are normally developed as stand-alone applications with a graphical user interface (GUI) that displays data via a range of controls. Rich client applications can be designed for both disconnected and connected scenarios (to cope with the need of occasionally accessing remote distributed data or functionality).

6.6.1.3 Web Application Architecture

A Web application can be accessed by the end users via a Web browser or a specialized user agent. The browser generates hypertext transfer protocol (HTTP) requests for specific uniform resource locators (URLs) that map to resources on a Web server which will render and return hypertext markup language (HTML) pages to the users. The core of Web application architecture lies in that it bolster connected scenarios and can support different browsers running on a wide range of operating systems and platforms.

6.6.2 Layered Structure

In the previous section, we have introduced distinct application architectures. One may notice that they all share one common structural settings, i.e. layers. So before we jump into our discussion regarding the components, let us first talk about layers. In general, layers represent the logical classifications of the functionality and the components within an application. This section introduces its attributes in terms of

internal (layer to layer) and external (layer to other clients or applications) communications. Typically, regardless of the type of applications (e.g. mobile and Web), we can always decompose the design into logical classifications of a set of software components. In practice, the introduction of layers makes it easier for us to distinguish between different tasks acted by separate components, which in turn helps to come up with a design that facilitates the reusability components and adaptability of applications. As we can see from previously introduced application architectures, each logical layer consists of a set of component types which are designed to perform discrete types of tasks.

By separating an application into distinct layers that have different roles and functionalities helps us to maximize the maintainability of the code, optimize the way that the application works when deployed in different scenarios, and offers a clear delineation between pinpoints where certain design decisions have to be made and specific technologies needs to be brought into play. Therefore, any application architecture, from the viewpoint of the highest and most abstract level, can be regarded as a set of collaborating components being classified into layers.

- *Perceiving layer*: The user-oriented functionality is usually contained in this layer for the purpose of managing user interaction with the system. Typically, this layer consists of components (see section below for more details) which serve as a common connection to the core action logic encapsulated in the acting layer.
- *Acting layer*: This layer, where the relevant action logic is encapsulated in, carries out the core functionality of the system. Different components are contained in this layer in which some of them can exhibit service interfaces that other callers may use.
- *Inferring layer*: This layer offers the means of approaching the data stored within the borders of the system and the data generated by other networked systems, say, different services.
- *Service layer*: From a high-level perspective, a hybrid software architecture can be seen as consisting of multiple services, each exchanging information with the others via transferring messages. Internally, each of these services is actually made up of software components. This layer effectively provides an alternative view that allows users to utilize various channels to access the application.

6.6.3 Building Block Components

Components offer a means of isolating specific groups of functionality within units that one can distribute and install separately from other functionality. This section covers some general details of creating components and discusses the types of components commonly utilized in each previously introduced layer types which can collectively form different application architectures.

6.6.3.1 Perceiving Layer Components

In general, the functionalities required for facilitating the interaction between users and the application itself are implemented by the perceiving layer components. The common used component types in this layer are as follows:

- *GUI (*i.e. *graphical user interface) components*: The specific user interface for the system is integrated into GUI components. These visual elements can be employed to convey message to users and accept users' input in the reverse order. In general, GUI components that represent the underlying data and logic within a system in an appropriate way, and to translate user input information and hand them to perception logic components where the principles of how the underlying data and application state are affected by various inputs, are defined. For the purpose of keeping a high level of adaptability, maintainability, and reusability, encapsulating ad hoc logic specifically in the GUI components is not recommended since such practice may hinder them from unit test.
- *Perception logic (PL) components*: Perception logic (PL) refers to the application code that defines the logical behaviour and structure of the perception. It is designed in a way that it is not dependent on any particular implementation of GUI components. The main responsibility of PL components is to orchestrate the users' interactions with the system by following a set of predefined rules. The existence of PL logic is independent of GUI components. In addition to this, PL components also take charge of transforming data from acting layer (see subsection below) into a usable format for GUI components. For instance, data from different sources may be aggregated and organized in such a way that they can be displayed more easily.

6.6.3.2 Acting Layer Components

The core functionalities of the system and the relevant action logic rules are normally implemented and encapsulated by acting layer components. The commonly used component types in this layer are as follows:

- *Application appearance (AA) components*: AA components generally offer a simplified interface to other action logic components, often by integrating multiple action operations into one operation so that action logic is easy to use and thus reducing dependencies since the components involved in acting layer and their inherent relationships do not necessarily need to be known by outside callers.
- *Action logic (AL) components*: AL can be regarded as any activity logic that is related to retrieving, processing, transforming, and managing application data, complying action rules and policies, and maintaining data consistency and validity. To maximize reuse or adaptation chances, AL components should not

contain any customized ad hoc logic elements. For the sake of convenience, we further divide the AL components into the following two subcategories:

- *Action flowchart (AF) components*: Once the user inputs are collected in perceiving layer and passed to the acting layer, the system can perform certain actions based on these inputs. Most acting processes involve multistage that should be executed in a reasonable order, though interactions with each other are often allowed. AF components work as coordinators for many complex, long-running, and multistep scenarios.
- *Action capsule (AC) components*: AC components encompass logic groups and data sets which are necessary to represent real-world elements, e.g. move forward or backward, within the target application domain. Particular data values are stored within AC and can be exposed later via properties. AC also serves as a container and manager of application data used by the systems.

6.6.3.3 Inferring Layer Components

The key aspect of inferring layer is to ensure the accuracy of the input data being collected. Inevitability, the type of data collected will directly impact the usefulness of the results, since the conclusions that can be drawn depend on how the data are collected. In addition, the smart environment will consist of numerous diverse sensor techniques that have their own data collection requirements. For example, on the one hand, devices like moisture sensors are simply broadcasting a state and generate only a few hundred bytes per day, while on the other hand, devices like remotely monitored mobile robots require a more complex model to capture useful data. In light of this statement, the data gathering requires some category description from the identification of data types to the frequency pattern of data published. Therefore inferring layer components provide access to two sources of data sets: First, data stored within the scope of system; second, data exposed by other networked systems. The commonly used component types in this layer are as follows:

- *Data processing (DP) components*: DP components abstract the logic required to access and process the underlying data sets. Several integrated functionalities can be found as below:
  - *Identification of data type*: The first step in the data processing is to identify the data types. The individuals in any particular population typically possess many characteristics. From the statist point of view, a variable is any characteristic whose value may change from one individual to another. The data resulted from making observations either on a single variable or simultaneously on two or more variables. In general, there are two types of data sets: categorical (or qualitative) and numerical (or quantitative), and the latter can be further classified in two different types: discrete and continuous. Also, the

collected data may be classified into four levels of scales of measurement, i.e. nominal scale, ordinal scale, interval scale, and ratio scale. Regarding the execution of data type identification, the use of graphs, charts, and tables is useful for understanding characteristics of the individual data attributes.

- *Discovery of published data frequency pattern*: Understanding the frequency pattern of published data is another main focus of data processing. For example, some data is dynamic, and others may need observation over time, implying learning and discovery.
- *Interpretation data*: Despite the seeking of input data precision, data relevance which according to management objectives is also a key subject. However, on the one hand, different symbols (e.g. letters, numbers, pictures, etc.) represent different facts, entities, or events. On the other hand, the data may contain errors or may have been collected in an inconsistent manner. Therefore, there is a pressing need for the development of data interpreting process to get valuable information. One common category of data interpreting tasks provides summarizations and statements about the data. Generally speaking, summarization is a process by which data is reduced for interpretation without sacrificing important information. In this category, data visualization, such as charts and summary tables, is an important tool. A second category focuses on the identification of important facts, relationships, anomalies, or trends in the data. Discovering this information often involves looking at the data in many ways using a rule-based analysis method. In general, a rule generally represents a decision in the form of "if. . .then. . ." proposition. The main goal of rule extraction is to discover hidden knowledge and explain it understandably, to extract previously unknown relations, and to ensure reasoning and defining capability. Among others, artificial neural network is one of the most widely used techniques for finding highly predictive rules. The third category of tasks involves the development of mathematical models that encode relationships in the data. In general, models can be built to predict continuous data values (e.g. regression models) or categorical data (e.g. classification models). The methods to generate these models include linear regression, logistic regression, neural networks, support vector machines, naïve Bayes, and different bio-inspired algorithm.
- *Filtering data*: In the data-filtering step, analysis is translated into the process of assessing at specific criteria to make it amenable to answering the problem outline at the beginning of the project, and hence this step should be carefully planned and executed. There are many ways to filter the results of data ranging from active filter, passive filter, to collaborative filter.
- *Storage data*: Finally, correct data is stored either in the local database or in the cloud.

- *Service agent (SA) components*: When the functionality provided by an external service must be introduced for an acting component to function, one might need to implement code to manage the semantics of communicating with that

particular service. This motivates us to design SA components who can isolate the issue of calling different services from the outside and can offer extra services, e.g. offline support, caching, etc.

6.6.3.4 Service Layer Components

The purpose of introducing service layer components is to provide other clients and applications with a way of access action logic in the application and take advantage of the functionality of the application via passing messages to and from it over a communication channel. In the proposed framework, we define two types of components, i.e. service interface (SI) components and message components. Their details are outlined as follows:

- *Service interface (SI) components*: Services reveal different SI to which all inbound messages are sent. An agreement is composed of the meaning of the set of messages forms which must be exchanged with a service so that the service can fulfil a specific action. SI can be treated as an appearance that exhibits the action logic implemented in the application to potential users.
- *Message components*: When transferring data via the service layer, data structures are normally wrapped by message structures that bolster various types of operations. These message components are the message agreements for communication between users and service providers.

6.6.3.5 Cross Layer Components

Cross layer components are designed to deal with the situation where many tasks implemented by the code of an application are needed in more than one single layer. Cross layer components carry out particular types of functionality that can be utilized by components from any layer. Typically, cross layer components consist of the following:

- *Security components*: These components are composed of those who can perform authentication-, authorization-, and validation-related tasks.
- *Management components*: This class includes components that can be operational tasks such as logging, configuration, tracking, and performance counters.
- *Communication components*: The components that can provide communication with other services and applications fall within this group.

6.7 Embedded Software for AAL

To offer more insights regarding how our SAfAAL would look like for embedded AAL devices, an example is elaborated in a smart home environment. In general, smart home system consists of multitudes of sensors and embedded equipment which generate large amounts of data. The abstract architecture is illustrated in Fig. 6.1.

As we can see from Fig. 6.1, the SAfAAL comprises four main layers, i.e. perceiving, acting, inferring, and service layer, plus one cross layer. In our model, five layers are physically distributed which is beneficial in terms of system's elasticity, adaptability, scalability, and extensibility. In addition, due to the diversity of AAL clients, the range of data observed from AAL can vary widely. Thus, we choose cloud computing to provide a real-time assisted-living service. Under this framework, adding or separating a new ALL client becomes no more difficult. Meanwhile, external service provider can join or leave the system freely. A brief overview of these layers and their corresponding responsibilities in the context of AAL is given below.

6.7.1 Smart Environment

Our proposed architecture can serve large amount of AAL clients. The specific settings of such smart environment are dependent on target user needs. Different components can be used in this layer for addressing discrete scenarios, say a set of clients including such as end user, a network of body sensors [28], and various other intelligent objects that form a smart home environment [64]. New elements can still be included without having to alter the architecture. Within such environment,

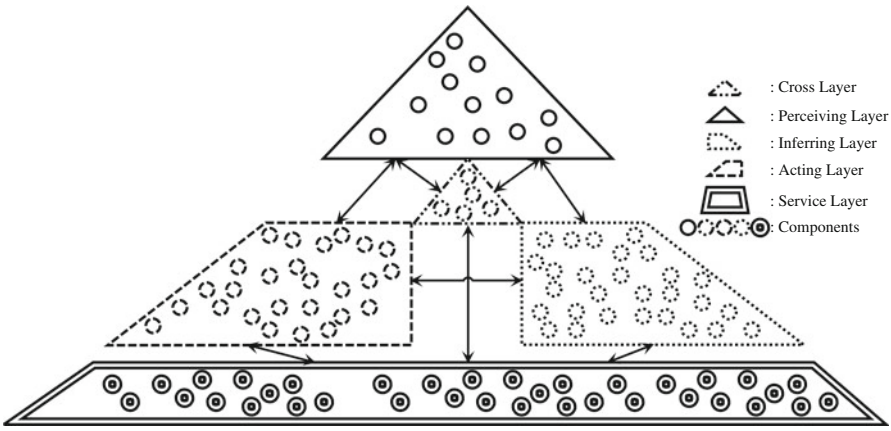


Fig. 6.1 System architecture of SAfAAL

different types of data can be perceived, observed, monitored, captured, and transported to the designated destination. Distinct GUIs displayed on mobile devices, smart objects, etc. carry out a real-time interaction with the end users. Although the complete setup depends on involved devices and the chosen service types, each part has a unique mapping component in perceiving layer. For instance, the perceived raw blood pressure data will be transferred to inferring layer for further processing. On the other hand, some flexible GUIs can also be used for providing useful visualized feedback to users. According to the literature [8, 53], the key configurations and infrastructure of the sensors that form a body sensor network often make them easily implemented. Nevertheless, due to their inherent characteristics such as low power life and wireless communication capability, the computation ability is often moved from these sensors to other layers for further treatment. When it is necessary, suitable communication protocols can be employed for calling them or further forwarding them to a more secured cloud.

6.7.2 Resort Presentation

Resort presentation is context aware-related applications. It can be seen as a software application running on various devices of the AAL client side. The function of resort presentation may include reminding doctor appointment, watching emergency, and recommending diet option. In other words, a set of rules and the associated list of assistive actions are integrated for responding to various observed conditions. Although the core principles hold for different types of actions, the detailed service content might be different. For instance, the treatment of the same sickness for a diabetes patient, for a food disorder patient, and for an obesity patient can be quite different. Due to the fact of the large number of action types, it is generally impracticable installing, deploying, and activating all the action components in a single layer. Only those scenario-related components are deployed first, while the others can be added in later. For a particular acting type, the similarities between context and action logic, a set of possible actions can be identified. As soon as a similar pattern is detected in the context, it is able to respond robust. In general all sorts of assistive actions are performed as outputs of this layer.

6.7.3 Data Interpretation

Making sense of data involves a lot of feature selection and classification tasks, and most of them are computationally expensive. This layer acts as one of the most important functionalities of the proposed model. The storing, retrieving, and process of context-related data are typically conducted within this layer. In practice, the fusion algorithm needs to be properly tuned to extract features from a large set of context for classification. Different DP components can be designed for

addressing common scenarios such as activity recognition, health monitoring, and fall detection. When extra data storage and memory are required for performing efficient classification, cloud service can be brought in for minimizing local servers' burden. The commonly used pattern recognition and classification algorithms are artificial neural networks, K-means clustering, Bayes classifiers, and decision trees [9, 69].

6.7.4 Context Aggregator and Providers

In this layer, contextual information can be aggregated in a single context model. In practice, a service provider can be a medical server which can calculate various medical data, or it can also be a weather station for forecasting context-related weather condition (e.g. temperature, humidity, etc.). Under these circumstances, a well-accepted context model, i.e. ontology-based context model [47], can be used to deal with ALL scenario. By using Web ontology language (OWL), the context model can be established around four main entities:

- *User ontology*: For identifying the user involved in the environment and his/her profile such as sex, age, diseases, health conditions, and social interactions
- *Location ontology*: For describing user's current position
- *Surround ontology*: For determining the conditions of user's surrounding environments which may potentially influence the assistive action decision-making
- *Equipment ontology*: For covering all the details of the sensors and other smart objects

The abovementioned abstraction is easily extendable and modifiable which in turn makes our architecture more adaptable. The introduction of this model can make the following two tasks easy: deriving new knowledge about the current context and detecting the extant inconsistency in the context data. Furthermore, it is also convertible to extensible markup language (XML), a flexible and platform-independent tool that can be employed in various stages of information representation, in the implementation level.

6.7.5 Cross Layer

Due to the nature of this layer, some commonly used functionalities, say, service management, key information access control, context-to-service mapping, complex computational tasks, are usually executed within this layer. For example, a software manager agent can be used to ensure the appropriate assistive services are delivered correctly and timely. Such context manager can automatically reconfigure its knowledge for adapting the system change. Meanwhile, security component is also one of the commonly used components within cross layer. Different security

solutions (e.g. [67] in the context of healthcare) can be integrated into distinct components to ensure the privacy of the services.

6.8 Trade-Off Strategy for Managing Adaptability

With adaptable software architecture, component modules can be added to a based software structure, and later actions (e.g. removal, rearrangement, or replacement) can be performed as required. Figure 6.2 illustrates a generalized adaptable software architecture.

As shown in Fig. 6.2, the software base component is located in the centre. Proceeding from the base are the components represented by circle with the letter "C". Different types of components are represented by various font types, and the empty circle denotes places where new components may be attached in the future. In order to keep the concept of an adaptable software architecture from exploding in complexity, boundaries can be imposed on the architecture via defining three basic architectural parameters, namely, component levels, component types, and component connectives.

Basically, component levels refer to the serial connections of components. For instance, three component levels are depicted in Fig. 6.2. The first level is composed of components which are attached to the software-based structure; the second level consists of components which are connected to the first-level components, and the remaining level(s) follows the same fashion. In general, it is possible to have any amount of component levels; however, possessing more component levels can no doubt increase the complexity of the software architecture and thus decrease its corresponding adaptability. Therefore, a trade-off strategy is often needed here for improving the software's usability.

As the second important architectural parameter, component types measure the number of unique components added on a machine. Conceivably, a software which

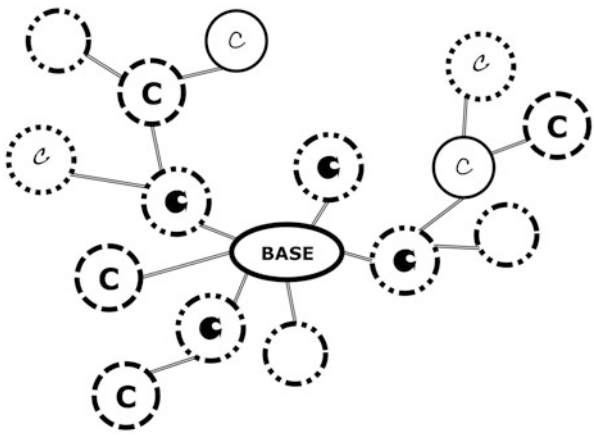


Fig. 6.2 Adaptable software architecture

has more component types may adapt to different utilization scenarios easily, although at a cost of with increased structure complexity degree.

The final parameter considered in this work refers to the number of available component connectives on the software base. These component connectives are connection points designated for accepting a component. For adaptability, the number of component connectives has the most immediate impact. If designed properly, an adaptable software architecture will increase in reusability with each additional or rearranged component. Hence, a machine with more component connectives will enjoy a greater reusability. Ideally, adaptable software architecture will include just enough component connectives to enable the necessary modification in capacity without significantly reducing software's reliability or increasing its corresponding maintaining cost.

6.8.1 Optimal Component Positions

The maximum number of component connectives allowed for an adaptable software architecture is a crucial design parameter. On one hand, it is a great advantage to develop a software architecture that could hold many components. The underlying reason is that the cost of adding components should be always less than starting again with another software architecture design. While on the other hand, with more components working together on a single software base, there is a high probability of software defaults. The result is the software's reliability becoming lower with each additional component being added. This generates a trade-off problem between the functionality advantages of additional components and usability losses due to the decreased reliability. This argument implies there is an existing optimal number of component connectives that can be included in a single adaptable software architecture. If more capacity is required above that point, a second software base should be introduced.

In order to decide the cost-optimal number of component connectives (N_{optimal}), we have to identify when the ratio of software usability rate to cost no longer increases with additional components. To proceed with the analysis, we introduce the following equations:

$$u_N = (N)\left(u_{\text{component}}\right) \tag{6.1}$$

Equation 6.1 gives the software usability rate of an adaptable software architecture (excluding software crashes), where $u_{\text{component}}$ denotes the usability rate of a component and u_N is the software usability rate as a function of N:

$$P_{\text{software}} = (P_{\text{base}})\left(P_{\text{component}}\right)^N \tag{6.2}$$

Equation 6.1 quantifies the effect of additional components on overall software reliability, where N is the number of components, $P_{\text{component}}$ refers to the probability of a component that is functional, and the probability of a software base that is able to function is represented by P_{base}.

If we combine Eqs. 6.1 and 6.2, Eq. 6.3 can be acquired. This is the usability rate of the software adjusted for availability:

$$u_N = (N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N \tag{6.3}$$

In practice (i.e. when $N > 0$ and $P_{\text{component}} < 1$), the usability rate function shown in Eq. 6.3 has a single maximum, which means one can determine the number of components ($N_{\max}$) that could offer the maximum usability rate. This can be done via firstly, differentiating u_N with respect to N and setting it equal to zero; secondly, solving the equation to get $N_{\max}$. The derivative of Eq. 6.3 is shown in Eq. 6.4 as follows:

$$\begin{aligned}\frac{\mathrm{du}_N}{\mathrm{dN}} &= \left[(N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\right]' \\ &= (N)'\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N + (N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left[\left(P_{\text{component}}\right)^N\right]' \\ &= \left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N + (N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left[\left(P_{\text{component}}\right)^N \ln P_{\text{component}}\right] \\ &= \left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\left[1 + (N)\ln P_{\text{component}}\right]\end{aligned} \tag{6.4}$$

After setting $\frac{\mathrm{du}_N}{\mathrm{dN}}$ equal to zero and calculating for $N_{\max}$, we can obtain Eq. 6.5 as follows:

$$N_{\max} = -\frac{1}{\ln P_{\text{component}}} \tag{6.5}$$

Equation 6.5 has an important implication since it shows the exact point when additional components will no longer increase the usability rate of the software. Nevertheless, this number ($N_{\max}$) is not necessarily equivalent to the cost-optimal number of component connectives (N_{optimal}). In fact, for most cases, the cost-optimal number will be lesser.

We then have to decide when the usability rate per component cost of an adaptable software architecture is maximized for the purpose of determining the cost-optimal number of components. After that point, a new software base should be introduced. In order to proceed with this calculation, we need to introduce an adaptable software architecture cost equation as shown in Eq. 6.6:

$$C_{\text{software}} = C_{\text{base}} + (N)C_{\text{component}} \tag{6.6}$$

where C_{base} is the cost of software base, $C_{\text{component}}$ represents the cost of a component, and C_{software} refers to the total cost of a software.

If u_N is divided by C_{software}, we get the ratio of usability rate to the cost of an adaptable software architecture with N components as shown in Eq. 6.7:

$$R_N = \frac{u_N}{C_{\text{software}}} = \frac{(N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N}{C_{\text{base}} + (N)C_{\text{component}}} \tag{6.7}$$

The optimum number of components (N_{optimal}) can occur when R_N reaches its maximum. Therefore, in the same manner as we calculate $N_{\max}$, we can get the derivative of R_N as shown in Eq. 6.8:

$$\begin{aligned}
\frac{\mathrm{dR}_N}{\mathrm{dN}} &= \left[\frac{(N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N}{C_{\text{base}} + (N)C_{\text{component}}}\right]' \\
&= \frac{\left[(N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\right]'\left[C_{\text{base}} + (N)C_{\text{component}}\right]}{\left[C_{\text{base}} + (N)C_{\text{component}}\right]^2} \\
&\quad - \frac{\left[(N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\right]\left[C_{\text{base}} + (N)C_{\text{component}}\right]'}{\left[C_{\text{base}} + (N)C_{\text{component}}\right]^2} \\
&= \frac{\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\left[1 + (N)\ln P_{\text{component}}\right]\left[C_{\text{base}}(N)C_{\text{component}}\right]}{\left[C_{\text{base}} + (N)C_{\text{component}}\right]^2} \\
&\quad - \frac{\left[(N)\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\right]C_{\text{component}}}{\left[C_{\text{base}} + (N)C_{\text{component}}\right]^2} \\
&= \frac{\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\left\{\left[1 + (N)\ln P_{\text{component}}\right]\left[C_{\text{base}} + (N)C_{\text{component}}\right] - (N)C_{\text{component}}\right\}}{\left[C_{\text{base}} + (N)C_{\text{module}}\right]^2} \\
&= \frac{\left(u_{\text{component}}\right)\left(P_{\text{base}}\right)\left(P_{\text{component}}\right)^N\left\{C_{\text{base}} + (N)\ln P_{\text{component}}\left[C_{\text{base}} + (N)C_{\text{component}}\right]\right\}}{\left[C_{\text{base}} + (N)C_{\text{component}}\right]^2}
\end{aligned} \tag{6.8}$$

After setting Eq. 6.8 equal to zero, we get the following quadratic equation as shown in Eq. 6.9:

$$(N)^2 \ln P_{\text{component}} C_{\text{component}} + (N)\ln P_{\text{component}} C_{\text{base}} + C_{\text{base}} = 0 \tag{6.9}$$

Finally, we can calculate N_{optimal} by solving Eq. 6.10:

$$\begin{aligned}
N_{\text{optimal}} &= \left\lfloor \frac{-\ln P_{\text{component}} C_{\text{base}} \pm \sqrt{\left(\ln P_{\text{component}} C_{\text{base}}\right)^2 - 4\left(\ln P_{\text{component}} C_{\text{component}}\right)C_{\text{base}}}}{2\ln P_{\text{component}} C_{\text{component}}} \right\rfloor \\
&= \left\lfloor \frac{-C_{\text{base}} \pm \sqrt{(C_{\text{base}})^2 - 4\left(\frac{C_{\text{component}}}{\ln P_{\text{component}}}\right)C_{\text{base}}}}{2C_{\text{component}}} \right\rfloor
\end{aligned} \tag{6.10}$$

where $\lfloor \cdot \rfloor$ means the floor of variable, i.e. rounding down to the nearest integer [65].

As we know, the $N_{optimal}$ must be an integer and also subject to the constraint of $N_{optimal} \geq 1$. So in practice, we have to use the positive root of the solution of Eq. 6.10 and truncate it to its nearest integer value.

6.9 Conclusion

During the last decade, AAL has gained in importance, since it can support elderly persons with their daily activities in order to help them maintain healthy and safety while living independently. However, most current models lack a consolidation view on how to manage those large-scale software-intensive systems. As a consequence, they are often limited to adaptability and interoperability. In this chapter, we propose a hybrid reference architecture for AAL, named software architecture for AAL (SAfAAL for short), which is built on the component-based architectural style. The ambitious goal is to balance the requirements and constraints imposed by the business, the ender users, and their surrounding environments. Typically, in software projects, software architectures reflect the major functional blocks of a software. They provide high-level abstractions that are helpful when software designers need to analyse the structure of whole system, e.g. finding a suitable balance between competing and potentially conflicting goals. Conventionally, these tasks were performed manually, assisted in human-based prior knowledge, and can be very time-consuming with the ever-increasing dimensionality of the software product. Fortunately, since the recent emergence of search-based software engineering (SBSE) [17, 33, 57], various tasks related to software architecture such as architectural optimization, components sequence identification, and component number determination can be formulated as complex search and optimization problems, where numerous computational intelligence techniques, both conventional (e.g. genetic algorithm [42], genetic programming [44], simulated annealing [45], Hill climbing [7]) and innovative (e.g. [76]), can certainly give us a hand. A number of studies have made several attempts in this regard [34, 41]; however, there are still a lot open questions that exist, for example, the definition of a representation of the problem, the fitness function, and the scalability of results [33]. Therefore, an immediate future research work can be done towards this direction.

References

1. Abowd GD, Mynatt ED (2004) Designing for the human experience in smart environments. In: Cook DJ, Das SK (eds) Smart environments: technology, protocols, and applications. Wiley, pp 153–174
2. Alemdar H, Ersoy C (2010) Wireless sensor networks for healthcare: a survey. Comput Netw 54:2688–2710

3. Andrade AO, Pereira AA, Walter S, Almeida R, Loureiro R, Compagna D, Kyberd PJ (2014) Bridging the gap between robotic technology and health care. Biomed Signal Process Control 10:65–78
4. Anonymous (2010) Ambient assisted living roadmap. European Ambient Assisted Living Innovation Alliance. IOS Press, Amsterdam
5. Anonymous (2014) Japan's demography: the incredible shrinking country. Economist (8889):35
6. Anonymous (2015) German demography. Economist (8929):56
7. Barros MO, Farzat FA, Travassos GH (2015) Learning from optimization: a case study with apache ant. Inf Softw Technol 57:1–8
8. Bellifemine F, Fortino G, Giannantonio R, Gravina R, Guerrieri A, Sgroi M (2011) SPINE: a domain-specific framework for rapid prototyping of WBSN applications. Softw Pract Exp 41:237–265
9. Bishop CM (2006) Pattern recognition and machine learning. Springer Science+Business Media, New York
10. Borja R, JRdl P, Álvarez A, Maestre JM (2013) Integration of service robots in the smart home by means of UPnP: a surveillance robot case study. Robot Auton Syst 61:153–160
11. Botia JA, Villa A, Palma J (2012) Ambient assisted living system for in-home monitoring of healthy independent elders. Expert Syst Appl 39:8136–8148
12. Bradford JW, Knott DG, Levine EH, Zemmel RW (2011) Accounting for the cost of U.S. health care: pre-reform trends and the impact of the recession. McKinsey & Company
13. Broadbent E, Stafford R, MacDonald B (2009) Acceptance of healthcare robots for the older population: review and future directions. Int J Soc Robot 1:319–330
14. Brodalski D, Brink H, Curtis J, Diaz S, Schindelar J, Shannon C, Wolfson C (2011) The health communicator's social media toolkit. Centers for Disease Control and Prevention (CDC), CS215469-A
15. Chernbumroong S, Cang S, Atkins A, Yu H (2013) Elderly activities recognition and classification for applications in assisted living. Expert Syst Appl 40:1662–1674
16. Cloutier R, Muller G, Verma D, Nilchiani R, Hole E, Bone M (2009) The concept of reference architectures. Syst Eng 13:14–27
17. Colanzi TE, Vergilio SR, Assunção WKG, Pozo A (2013) Search based software engineering: review and analysis of the field in Brazil. J Syst Softw 86:970–984
18. Cooper RA, Grindle GG, Vazquez JJ, Xu J, Wang H, Candiotti J, Chung C, Salatin B, Houston E, Kelleher A, Cooper R, Teodorski E, Beach S (2012) Personal mobility and manipulation appliance-design, development, and initial testing. Proc IEEE 100:2505–2511
19. Díaz M, Juan G, Lucas O, Ryuga A (2012) Big data on the Internet of things: an example for the E-health. In: Sixth international conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), IEEE, Palermo, 4–6 July, pp 898–900
20. Dimitrov T (2005) Design and implementation of a home automation service gateway based on OSGi. University of Duisburg-Essen, Düsseldorf
21. Ding D, Cooper RA, Pasquina PF, Fici-Pasquina L (2011) Sensor technology for smart homes. Maturitas 69:131–136
22. Doukas C, Metsis V, Becker E, Le Z, Makedon F, Maglogiannis I (2011) Digital cities of the future: extending @home assistive technologies for the elderly and the disabled. Telematics Inform 28:176–190
23. Eklund U, Askerdal Ö, Granholm J, Alminger A, Axelsson J (2005) Experience of introducing reference architectures in the development of automotive electronic systems. SIGSOFT Softw Eng Notes 30:1–6
24. Falconer J (2013) HOSPI-R drug delivery robot frees nurses to do more important work. http://www.gizmag.com/panasonic-hospi-r-delivery-robot/29565/. Accessed on 30 July 2015
25. Feil-Seifer D, Matarić MJ (2005) Defining socially assistive robotics. In: Proceedings of the 2005 I.E. 9th international conference on rehabilitation robotics, Chicago, 28 June–1 July 2005

26. Fernández-Montes A, Ortega JA, Sánchez-Venzalá JI, González-Abril L (2014) Software reference architecture for smart environments: perception. Comp Stand Interfaces 36:928–940
27. Flammini A, Sisinni E (2014) Wireless sensor networking in the internet of things and cloud computing era. Prod Eng 87:672–679
28. Fortino G, Giannantonio R, Gravina R, Kuryloski P, Jafari R (2013) Enabling effective programming and flexible management of efficient body sensor network applications. IEEE Trans Hum-Mach Syst 43:115–133
29. Fujiwara N, Hagiwara Y, Choi Y (2012) Development of a learning support system with PaPeRo. The 12th international conference on control, automation and systems, Jeju Island, 17–21 Oct, pp 1912–1915
30. Graf B, Hans M, Schraft RD (2004) Care-O-bot II: development of a next generation robotic home assistant. Auton Robot 16:193–205
31. Graf B, Parlitz C, Hägele M (2009) Robotic home assistant Care-O-bot 3 product vision and innovation platform. In: Jacko JA (ed) Human-computer interaction, part II, (HCII 2009), LNCS 5611. Springer, Berlin, pp 312–320
32. Graf B (2014) Care-O-bot. Fraunhofer Institute for Manufacturing Engineering and Automation. http://www.care-o-bot.de/en/care-o-bot-3.html. Accessed 30 July 2015
33. Harman M, Mansouri SA, Zhang Y (2012) Search based software engineering: trends, techniques and applications. ACM Comput Surv 45:1101–1164
34. Harman M, Lakhotia K, Singer J, White DR, Yoo S (2013) Cloud engineering is search based software engineering too. J Syst Softw 86:2225–2241
35. Harrefors C, Axelsson K, Sävenstedt S (2010) Using assistive technology services at differing levels of care: healthy older couples' perceptions. J Adv Nurs 66:1523–1532
36. HITACHI (2014) Robotics: EMIEW 2. http://www.hitachi.com/rd/portal/research/robotics/emiew2_01.html. Accessed 30 July 2015
37. Hong YJ, Kim IJ, Ahn SC, Kim HG (2010) Mobile health monitoring system based on activity recognition using accelerometer. Simul Model Pract Theory 18:446–455
38. Hosoda Y, Egawa S, Tamamoto J, Yamamoto K, Nakamura R, Togami M (2006) Basic design of human-symbiotic robot EMIEW. Proceedings of the 2006 IEEE/ RSJ international conference on intelligent robots and systems,9–15 Oct, Beijing, pp 5079–5084
39. Hosoda Y, Yamamoto K, Ichinose R, Egawa S, Tamamoto J (2010) Collision-avoidance algorithm for human-symbiotic robot. International conference on control, automation and systems 2010, 27–30 Oct, Gyeonggi-do, pp 557–561
40. Hydra Project (2011) Hydra open source middleware
41. Kempka J, McMinn P, Sudholt D (in press) Design and analysis of different alternating variable searches for search-based software testing. Theor Comput Sci
42. Konak A, Coit DW, Smith AE (2006) Multi-objective optimization using genetic algorithms: a tutorial. Reliab Eng Syst Saf 91:992–1007
43. Kuindersma SR, Hannigan E, Ruiken D, Grupen RA (2009) Dexterous mobility with the uBot-5 mobile manipulator international conference on advanced robotics (ICAR), pp 1–7
44. Langdon WB, Poli R, McPhee NF, Koza JR (2008) Genetic programming: an introduction and tutorial, with a survey of techniques and applications. Stud Comput Intell (SCI) 115:927–1028
45. Matinnejad R, Nejati S, Briand L (2015) Search-based automated testing of continuous controllers: framework, tool support, and case studies. Inf Softw Technol 57:9–15
46. McGrath MJ, Scanaill CN (2014) Sensor technologies: healthcare, wellness, and environmental applications. Apress Media, LLC, New York, ISBN 978-1-4302-6013-4
47. Mocholí J, Sala P, Fernández-Llatas C, Naranjo J (2010) Ontology for modeling interaction in ambient assisted living environments. In: The 15th Mediterranean conference on medical and biological engineering and computing, Springer, pp 655–658
48. Mukai T, Hirano S, Nakashima H, Kato Y, Sakaida Y, Guo S, Hosoe S (2010) Development of a nursing-care assistant robot RIBA that can lift a human in Its arms. In: The 2010 IEEE/RSJ international conference on intelligent robots and systems, 18–22 Oct 2010, Taipei

49. Murai R, Sakai T, Kawano H, Matsukawa Y (2012) A novel visible light communication system for enhanced control of autonomous delivery robots in a hospital. IEEE/SICE international symposium on System Integration (SII), 16–18 Dec, Kyushu University, Fukuoka, pp 510–516
50. Nakagawa EY, Antonino PO, Becker M, Maldonado JC, Storf H, Villela KB, Rombach D (2013) Relevance and perspectives of AAL in Brazil. J Syst Softw 86:985–996
51. OASIS Project (2011) OASIS: quality of life for the elderly
52. OpenAAL (2011) OpenAAL. The open source middleware for ambient-assisted living
53. Otto C, Milenkovic A, Sanders C, Jovanov E (2006) System architecture of a wireless body area sensor network for ubiquitous health monitoring. J Mob Multimed 1:307–326
54. Paschou M, Sakkopoulos E, Sourla E, Tsakalidis A (2013) Health internet of things: metrics and methods for efficient data transfer. Simul Model Pract Theory 34:186–199
55. Pekka P, Pakkala D (In press) Reference architecture and classification of technologies, products and services for big data systems. Big Data Res
56. PERSONA Project (2011) Perceptive Spaces promoting independent aging
57. Räihä O (2010) A survey on search-based software design. Comp Sci Rev 4:203–249
58. Rashidi P, Mihailidis A (2013) A survey on ambient-assisted living tools for older adults. IEEE J Biomed Health Informa 17:579–590
59. RIKEN-TRI Collaboration Center RIBA. http://rtc.nagoya.riken.jp/RIBA/index-e.html. Accessed on 30 July 2015
60. Sato M, Sugiyama A, Ohnaka Si (2006) Auditory system in a personal robot, PaPeRo. In: 2006 digest of technical papers international conference on consumer electronics (ICCE 06), 7–11 Jan 2006, pp 19–20
61. Sato M, Iwasawa T, Sugiyama A, Nishizawa T, Takano Y (2009) A single-chip speech dialogue module and its evaluation on a personal robot, PaPeRo-mini. In: IEEE international conference on acoustics, speech and signal processing (ICASSP), 19–24 Apr, Taipei, pp 3697–3700
62. Schmidt RF (2013) Software engineering: architecture-driven software development. Morgan Kaufmann, Elsevier, Waltham, ISBN 978-0-12-407768-3
63. Shen VRL, Yang C-Y, Chen CH (2015) Smart home management system with hierarchical behavior suggestion and recovery mechanism. Comp Stand Interfaces 41:98–111
64. Spivey D (2015) Home automation for Dummies. Wiley, Hoboken, ISBN 978–1–118-94926-907030–5774
65. Stewart J (2012) Calculus. Brooks/Cole, Cengage Learning, Belmont. ISBN 978-0-538-49781-7
66. Su C-J, Chiang C-Y (2014) Pervasive community care platform: ambient intelligence leveraging sensor networks and mobile agents. Int J Syst Sci 45:778–797
67. Sun J, Zhu X, Zhang C, Fang Y (2012) Security and privacy for mobile health-care (m-Health) systems. In: Das SK, Kant K, Zhang N (eds) Handbook on securing cyber-physical critical infrastructure. Elsevier, Waltham, pp 677–704. ISBN 978-0-12-415815-3
68. Suryadevara NK, Mukhopadhyay SC, Wang R, Rayudu RK (2013) Forecasting the behavior of an elderly using wireless sensors data in a smart home. Eng Appl Artif Intell 26:2641–2652
69. Theodoridis S, Koutroumbas K (2009) Pattern recognition. Academic (Elsevier)
70. UniversAAL Project (2011) The universAAL reference architecture
71. Vance K, Howe W, Dellavalle RP (2009) Social internet sites as a source of public health information. Dermatol Clin 27:133–136
72. Vandewynckel J, Otis M, Bouchard B, Ménélas B-A-J, Bouzouane A (2013) Towards a real-time error detection within a smart home by using activity recognition with a shoe-mounted accelerometer. Procedia Comp Sci 19:516–523
73. Wang H, Grindle GG, Candiotti J, Chung C, Shino M, Houston E, Cooper RA (2012) The personal mobility and manipulation appliance (PerMMA): a robotic wheelchair with advanced mobility and manipulation. In: The 34th annual international conference of the IEEE EMBS, San Diego, 28 Aug−1 Sept 2012

74. Wu Y-C, Chang C-S, Sawaguchi Y, Yu W-C, Chen M-J, Lin J-Y, Liu S-M, Han C-C, Huang W-L, Su C-Y (2011) A mobile-phone-based health management system. In: Smigorski K (ed) Health management – different approaches and solutions. InTech, Rijeka, ISBN 978–953–307-296-8
75. Wu Y-H, Wrobel J, Cristancho-Lacroix V, Kamali L, Chetouani M, Duhaut D, Pevedic BL, Jost C, Dupourque V, Ghrissi M, Rigaud A-S (2013) Designing an assistive robot for older adults: the ROBADOM project. IRBM 34:119–123
76. Xing B, Gao W-J (2014) Innovative computational intelligence: a rough guide to 134 clever algorithms. Springer International Publishing Switzerland, Cham. ISBN 978-3-319-03403-4
77. Xu J, Grindle GG, Salatin B, Vazquez JJ, Wang H, Ding D, Cooper RA (2010) Enhanced bimanual manipulation assistance with the personal mobility and manipulation appliance (PerMMA). The 2010 IEEE/RSJ international conference on intelligent robots and systems, 18–22 Oct 2010, Taipei
78. Zhang J, Shan Y, Huang K (2015) ISEE smart home (ISH): smart video analysis for home security. Neurocomputing 149:752–766

Chapter 7
3PR Framework for Software Project Management: People, Process, Product, and Risk

Kadir Alpaslan Demir

7.1 Introduction

There are various studies reporting the success and failure rates in software projects [11, 13, 48]. Even with the lowest failure rates reported, software projects are significantly failing when compared to projects in other fields. Slevin and his colleagues [49] identified a number of project management issues in several project-based industries. Among nine industries studied, only for the software industry, poor performance is explicitly reported as an issue. The average software project is likely to be 6–12 months behind schedule and 50–100% over budget [50]. Ineffective software project management is one of the main reasons for software project failures [51]. In addition, effective project management is an important factor in achieving software project success [10, 51]. DeMarco and Lister [26] state: "For overwhelming majority of the bankrupt projects we studied, there was not a single technological issue to explain the failure." Robertson and Robertson emphasize that, "in several decades of project experience, we have never seen a project fail for technical reasons. It has always been human failures that have caused otherwise good projects grind to a halt" [53]. Various other researchers and practitioners emphasize the importance of software project management in the success and failure of software projects [27, 54]. According to Boehm, poor management may increase software costs more rapidly than any other factor [55]. COCOMO, a method for software project cost and effort estimation developed by Barry Boehm and his colleagues, does not include project management as a factor [55]. Therefore, in COCOMO II, the estimation model incorporates some

K.A. Demir (✉)
Department of Software Development, Turkish Naval Research Center Command, Istanbul, Turkey
e-mail: kadiralpaslandemir@gmail.com

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_7

project management-related factors such as PCON (personnel continuity) and PMAT (process maturity) [56].

One would expect that our performance in software projects should have been better with all the advancements in technical aspects of software engineering. However, relying merely on technological advances to achieve better outcomes in software projects may be misleading. Significant advances in software project management are also required. Therefore, proposals and discussions for applicable and viable theories, models, and practices in software project management are important steps in achieving better project outcomes. Furthermore, we need better project management software tools, since these tools are essential for effective project management [52].

We achieved a certain level of success in various types of projects. Based on the experience gained over the years, Project Management Institute (PMI) developed the infamous standard "Project Management Body of Knowledge (PMBOK Guide)." There are also many studies focusing on project management especially identifying critical success factors and causes of failures in projects. The current low success rates in software projects calls for more research focusing on software project management. Cooke-Davies identified that different types of projects require different project management practices [28]. As a result, we need to continue searching for better theories, models, and different approaches for software project management. The goal of this study to develop a software project management framework. The framework is called 3PR (people, process, product, risk) framework for software project management. It is developed as a result of an extensive literature review of project management and software engineering literature. 3PR framework is validated with an international survey of software practitioners. The framework is used to develop a metric for software project management effectiveness [10]. The 3PR framework is comprehensive, and it can be used as a basis for many other research studies on software project management.

This chapter outlines the 3PR framework. First, an overview of project management approaches, models, and frameworks is presented. Then, we describe the framework and the survey study conducted to validate the framework before finalizing the chapter.

7.2 Software Project Management Approaches, Models, and Frameworks

In this section, we review the leading approaches, models, and frameworks for software project management. First, let's review a few project management models and frameworks.

Currently, the established standard for project management is Project Management Institute's (PMI) "A Guide to the Project Management Body of Knowledge"

(PMBOK Guide). Hundreds of project management professionals contributed to the development of PMBOK Guide over time. It embodies the cumulative project management experience gained throughout the years. It is a general guide for project management. Therefore, it can be used in projects from various industries. In addition, the PMBOK Guide has extensions for software, construction, and government projects. The first edition [1] was published in 1996. The latest edition is the fifth edition [5] published in 2013. PMBOK Guide identifies five project management process groups [1–5] as follows:

- Initiating process group
- Planning process group
- Executing process group
- Monitoring and controlling process group
- Closing process group

According to the PMBOK, these are not the phases of a project, and they may be repeated for each phase where appropriate. PMBOK also identifies and lists ten project management knowledge areas [5]:

1. Project integration management
2. Project scope management
3. Project time management
4. Project cost management
5. Project quality management
6. Project human resource management
7. Project communications management
8. Project risk management
9. Project procurement management
10. Project stakeholder management

One notable change in the last edition is the addition of project stakeholder management as a separate knowledge area. In the previous editions, it was a part of project communications management. The importance of stakeholder management is recognized by separating it as another knowledge area.

According to PMBOK Guide [5], there are organizational influences on the management of a project. Organizational characteristics, factors, and assets create these influences. Organizational cultures and styles, organizational communications, organizational structures, organizational process assets, and enterprise environmental factors are among these organizational characteristics, factors, and assets. PMBOK Guide also lists key interpersonal management skills for project managers. These are leadership, team building, motivation, communication, influencing, decision-making, political and cultural awareness, negotiation, trust building, conflict management, and coaching.

Overall, PMBOK Guide identifies 47 project management processes. These processes are categorized under five process groups and ten knowledge areas. The guide has a process-oriented view on project management. Projects are managed as a set of processes. While PMBOK Guide points out the importance of

organizational influences and various management skills, interactions between processes and management elements are not detailed in the guide.

Forsberg [47] developed an elegant model for project management. The model is called "the wheel and axle model." It is based on five essentials for every project:

- Organizational commitment
- Communication
- Teamwork
- Project cycle
- Management elements

In the wheel and axle model, organizational commitment, communication, and teamwork support the whole project development cycle and key management elements. The project development cycle has three aspects: business, budget, and technical. These aspects should be kept in balance [47]. In the model, a project has three periods and each period contains a number of phases. The defined periods in the wheel and axle model are study period, implementation period, and operations period. One of the five essentials is management elements. These management elements are [47]:

- Project requirements
- Organizational options
- Project team
- Project planning
- Opportunities and risks
- Project control
- Project visibility
- Project status
- Corrective actions
- Project leadership

According to Forsberg, Mooz, and Cotterman [47], project leadership binds the other nine management elements. These researchers place a special emphasis on project leadership. This emphasis is right on target, since our research findings indicate that project managers and project leaders at every level play a crucial role in project success [10]. Furthermore, people management is essential in the model. The interactions between project management processes and key management elements are detailed in the wheel and axle model. In conclusion, Forsberg, Mooz, and Cotterman have a different approach than the PMBOK Guide's process-oriented approach. They combine project management processes, project development cycle, and management elements in their model. These researchers provide a holistic view on project management. Their book on project management is based on this wheel and axel model of project management [47].

We briefly discussed two different approaches to project management. These approaches are applicable to projects from different industries. Now, let us focus on the approaches and models for software project management.

Capability Maturity Models (CMMs) developed by Carnegie Mellon Software Engineering Institute (SEI) had a significant impact on the software industry. Basically, CMMs establish organizational maturity levels for software development and related areas. There are five organizational maturity levels: initial, managed, defined, quantitatively managed, optimizing. CMMs identify required processes and activities for each maturity level. The assumption is that mature organizations develop systems better. Many organizations got CMM certifications. Furthermore, numerous organizations adapted the processes detailed in CMMs. In the beginning of 1990s, first CMMs were released. Later, in 2002, SEI published Capability Maturity Model Integration (CMMI) Version 1.1 [7]. CMMI combines a number of earlier CMMs under one model. The focus in CMMI is the organization rather than the project.

CMMI for Development Version 1.3 (CMMI-DEV, V1.3) [9] describes four main process areas: process management, project management, engineering, and support. These are further divided into basic and advanced process areas.

Process management process areas include the following five process areas:

- Organizational process definition (OPD)
- Organizational process focus (OPF)
- Organizational performance management (OPM)
- Organizational process performance (OPP)
- Organizational training (OT)

Organizational process definition, organizational process focus, and organizational training are among the basic process areas. Organizational performance management and organizational process performance are advanced process areas.

Project management process areas include the following seven process areas:

- Integrated project management (IPM)
- Project monitoring and control (PMC)
- Project planning (PP)
- Quantitative project management (QPM)
- Requirements management (REQM)
- Risk management (RSKM)
- Supplier agreement and management (SAM)

Four of these areas are basic project management process areas. These are project planning, project monitoring and control, requirements management, and supplier agreement and management. Advanced project management process areas are composed of integrated project management, quantitative project management, and risk management. Engineering process areas includes the following five process areas:

- Product integration (PI)
- Requirements development (RD)
- Technical solution (TS)
- Validation (VAL)
- Verification (VER)

Support process areas include the following five process areas:

- Causal analysis and resolution (CAR)
- Configuration management (CM)
- Decision analysis and resolution (DAR)
- Measurement and analysis (MA)
- Process and product quality assurance (PPQA)

Process and product quality assurance, configuration management, and measurement and analysis process areas are basic support process areas. Causal analysis and resolution and decision analysis and resolution process areas are categorized under advanced support process areas.

Note that project management process area has more process areas than others have. Apparently, CMMI recognizes the importance of project management in system and software developments. Furthermore, some of the process areas from other main areas are also closely related to project management. For example, measurement and analysis, decision analysis and resolution, and almost all process areas in process management are related to project management.

SEI also developed a People Capability Maturity Model (P-CMM). The P-CMM is a set of human capital management processes helping to improve the capability of an organization's workforce [65]. Since CMMI omits people management issues, P-CMM compliments CMMI in this aspect. Like PMBOK Guide, CMMI has a process-oriented approach to systems development and project management.

The "Program Manager's Guide to Software Acquisition Best Practices Version 2.31" was prepared by the Software Program Managers Network (SPMN). The network consisted of many experienced software program managers. The guide identifies nine principal best practices [6] as follows:

- Formal risk management
- Agreement on interfaces
- Formal inspections
- Metrics-based scheduling and management
- Binary quality gates at the inch-pebble level
- Program-wide visibility of progress vs. plan
- Defect tracking against quality gates
- Configuration management
- People-aware management accountability

Also, the guide groups the best practices into seven proven management areas as shown below:

- Risk management
- Planning
- Program visibility
- Program control
- Engineering practices and culture
- Process improvement
- Solicitation and contracting

Every management area contains a number of best practices. For example, risk management has five best practices, planning has four practices, program visibility has four practices, and so on. The "Program Manager's Guide to Software Acquisition Best Practices Version 2.31" encompasses many essentials of software program management. Most of the practices overlap with PMBOK Guide and CMMI. However, unlike PMBOK Guide or CMMI, this guide does not have a structured approach for project management.

The Software Quality Institute's Body of Knowledge for Software Project Management (SQI BOK) lists 34 competencies. This list of essential competencies is employed by the most successful software project managers. These competencies are categorized into three parts, product, project, and people [46], as mentioned below.

Product Development Techniques

1. Assessing processes
2. Awareness of process standards
3. Defining the product
4. Evaluating alternative processes
5. Managing requirements
6. Managing subcontractors
7. Performing the initial assessment
8. Selecting methods and tools
9. Tailoring processes
10. Tracking product quality
11. Understanding development activities

Project Management Skills

12. Building a work breakdown structure
13. Documenting plans
14. Estimating cost
15. Estimating effort
16. Managing risks
17. Monitoring development
18. Scheduling
19. Selecting metrics
20. Selecting project management tools
21. Tracking processes
22. Tracking project progress

People Management Skills

23. Appraising performance
24. Handling intellectual property
25. Holding effective meetings
26. Interaction and communication
27. Leadership

28. Managing change
29. Negotiating successfully
30. Planning careers
31. Presenting effectively
32. Recruiting
33. Selecting a team
34. Teambuilding

SQI BOK has a competency-based approach for software project management. It does a good job in capturing the required competencies. The project managers and leaders at various levels should be competent in some or all of these competencies. As a result, leaders with these competencies are assumed to do well in completing a project successfully.

Philips identifies three key perspectives for software project management: people, business, and process [12]. He emphasizes that having these perspectives won't make a project successful, but it will help to go a long way to making success possible. He promotes four basic principles that need to be applied with discipline and perseverance viz.:

- Balance people, process, and product
- Promote visibility
- Organize by using configuration management tools properly
- Use standards judiciously

Philips highlights that all undertakings include the 3Ps: people, process, and product [12]. In successful undertakings, these 3Ps are managed in harmony.

Wiegers [14] identifies five dimensions of a software project: staff, features, quality, cost, and schedule. Throughout a software project, the listed five dimensions have to be managed. Figure 7.1 shows these dimensions. These dimensions are somewhat dependent on each other; the relations among them are nonlinear and complex most of the time. The dimensions may be assigned roles on a project: a driver, a constraint, or a degree of freedom [14]. The driver of a project is the key objective. There may be multiple drivers. However, if all dimensions are assumed to be drivers, there is no point in having different roles. A constraint is the limiting factor for the project. The constraint has to be outside of the project manager's control. For example, a fixed cost price, where negotiation with the customer is not an option, is the constraint. When the team size is fixed and the manager is not allowed to hire new team members or detach team members from the project organization, then the staff is the constraint. The rest of the dimensions that are not drivers or constraints become the degrees of freedom. When the project manager has control over adding or omitting features, then the feature dimension is a degree of freedom.

Figure 7.1 presents the dimensions on a Kiviat diagram. Kiviat diagrams are useful when multiple item evaluations are presented on a single diagram. A Kiviat diagram is a polygon, which has the same number of sides as the number of variables. Each axis represents a data category, and different scales and data

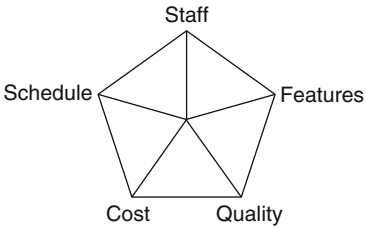


Fig. 7.1 Five dimensions of a project

types can be used. However, in this case, the same scale is used. The dimensions are categorized with respect to the flexibility the project manager has over the dimension. The flexibility of the dimension is plotted on an axis of the Kiviat diagram. The scale on a dimension goes from zero flexibility to highest flexibility (0–10). The closer the plot is to the center, the less flexibility there is for that dimension. So, for a complete constraint such as having a fixed number of team members, the plot on the staff axis would be the closest to the center. Understanding the driver, the constraint, and the degrees of freedom in a project and plotting them on a Kiviat diagram help us in critical decision-making as well as with prioritization.

According to Bach, all managers are faced with the 3Ps while developing software [15]. These 3Ps are people, problem, and process. He questions whether the 3Ps should be given equal weight and whether one should be given more focus than others. Bach emphasizes that the people aspect of software development should be given more focus. He criticizes CMM for focusing too much on process rather than people at the time. One year later in 1995, the first version of the People Capability Maturity Model (P-CMM) [16] was released based on the work by Humphrey [17]. Later the work was called Personal Software Process [18, 19].

Kulpa reports an interesting graphic from a CMM introduction class [20]. The graphic presents the foundations for an organization and consists of a three-legged stool figure. In the graphic, the stool represents the organization. The legs of the stool are people, process, and technology. Kulpa argues that a stable organization needs these three legs. While most organizations try to improve their processes and implement various technologies, they often neglect the people aspect. Kulpa argues that after a CMMI-based process improvement effort, implementing People CMM should complement this effort. Kulpa basically stresses the importance of people management in an organization.

PMBOK Guide and CMMI are the results of extensive work. Both are supported by well-established institutions. They are developed with the help of numerous researchers and practitioners. Both PMBOK Guide and CMMI are process oriented and they have a systematic structure. While they recognize the importance of people management, they do not go into the details of people management in the project environment. The Program Manager's Guide to Software Acquisition Best Practices Version 2.31 and the Software Quality Institute's Body of Knowledge for Software Project Management are quite comprehensive. They provide a different approach to project management. One focuses on best practices, while the

other focuses on project management competencies. We also briefly discussed some other work on project management models, approaches, and perspectives. Most of these provide a general perspective to project management and focus on some essential project management areas. These works are generally based on the experiences of the respective researchers.

7.3 3PR (People, Process, Product, and Risk) Framework for Software Project Management

The goal of 3PR (people, process, product, and risk) framework research was to develop a simple project management framework customized for software project management. This research was conducted as part of a bigger research project. The goal of the bigger research project was to develop a software project management effectiveness metric [10]. Such a metric development study required a valid software project management framework. We investigated the project management frameworks and models in the current literature. Some of these are already discussed in the previous section. We concluded that the current frameworks and models are insufficient for guiding the development of a comprehensive software project management effectiveness metric. Some of the previous models and frameworks are process oriented, and people management is somewhat overlooked. Some provide a holistic view without adequate details. Some are arguably complete. Most importantly, most of these studies need validation. As a result, we developed the 3PR framework and validated it. This framework was successfully used as a foundation for the development of a software project management effectiveness metric [10].

The 3PR framework consists of four main areas in software project management:

- People management
- Process management
- Product management
- Risk management

The first letters of main software project management areas are combined, and the framework is named as 3PR. The framework is shown in Fig. 7.2. In the following sections, we briefly discuss each software project management area.

7.3.1 People Management

The importance of people management in project development efforts is quite well established [12, 15, 16, 19, 21–26]. The people management area is especially

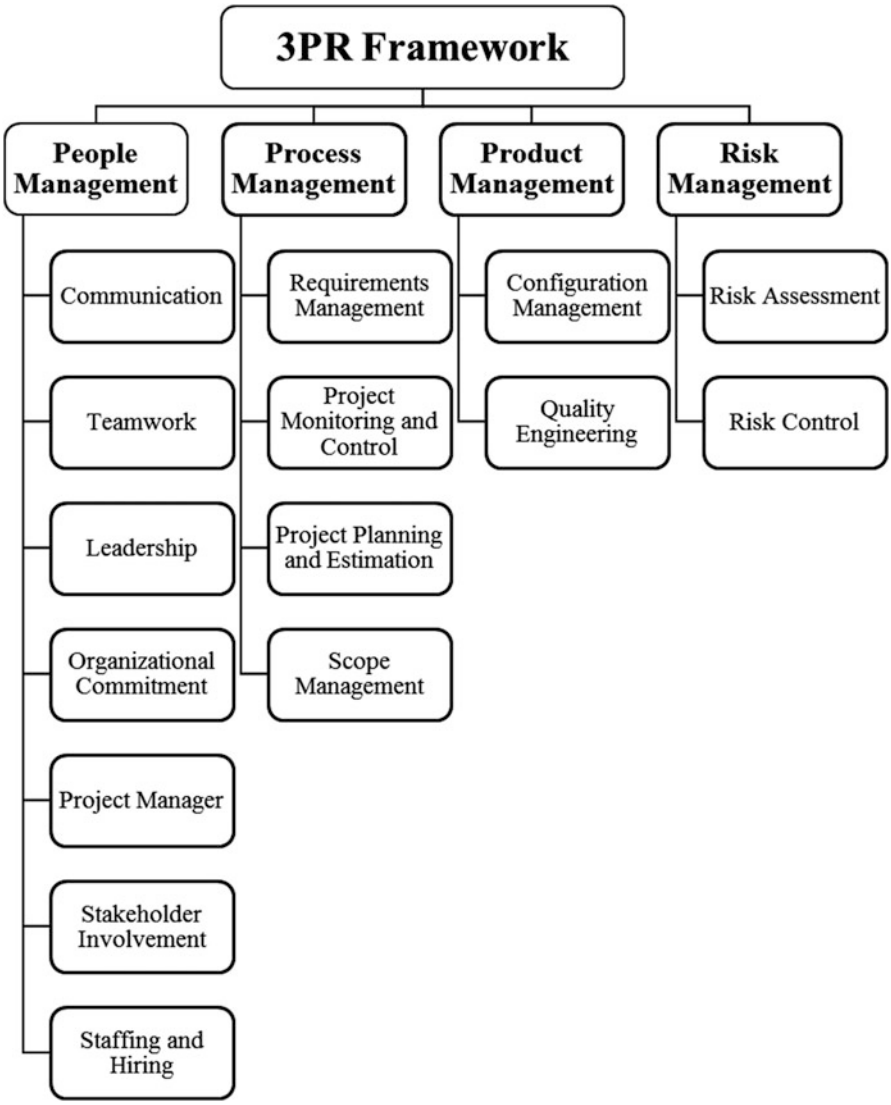


Fig. 7.2 3PR framework for software project management

important in software development projects, since the development is considerably human intensive compared to other industries.

James Bach takes a radical position in what aspect needs more focus in software development projects [15]. He strongly points out that "At conferences and in journals, the extraordinary attention we give to software development processes is misplaced. Far too much is written about processes and methods for developing software; far too little about care and feeding of the minds that actually write the

software. Process is useful, but it is not central to successful software projects. The central issue is the human processor – the hero who steps up and solves the problems that lie between a need expressed and a need fulfilled." He also emphasizes that "I argue that the only basis for success of any kind is the heroic efforts of a dedicated team." Even though his views might be seen as radical, this may be the result of resentment due to lack of research and emphasis on people issues in software development when compared to research on processes. Weinberg states, "The three causes of failure are people, people, and people" [27]. Again, Thomsett points out that "most projects fail because of people and project management concerns rather than technical issues" [57]. Kulpa states that the one area that is unaddressed by organizations is the people [20]. Philips takes a more central approach [12]. He stresses the importance of having a balance between people, process, and product. He argues that the road to success passes from harmonizing these 3Ps.

Brooks pointed out the variations in programmer productivity as a problem [21]. He references studies reporting an order of magnitude variations dated back to 1968 [29]. DeMarco and Lister reported significant computer programmer productivity variations ranging from one to tenfold [25]. Weinberg reported variations in programmer productivity and quality from 20 to 1 [27]. Considerable variations exist in software development productivity. Measuring programmer productivity is not trivial. It is hard to set up an experiment in which it is possible to control every factor contributing to the productivity and to measure the productivity correctly.

In one of the most widely known cost estimation technique, COCOMO II, team cohesion affects the effort estimation exponentially. The team cohesion scale factor accounts for the difficulties in managing different stakeholders including users, customers, developers, etc. [30].

Hughes and Cotterell [31] point out that people with practical experience in software projects will clearly declare people management as an important aspect of software project management.

The importance of people management in software development projects is established. Therefore, people management area is essential in the 3PR framework. The study conducted for the validation of the framework also shows that people management has the highest importance in software project management.

7.3.2 Process Management

Without a defined process, gathering a bunch of practitioners and expecting them to work in harmony for a common goal is very unlikely. Two things may happen: either they naturally form a team through group dynamics and even set up a process invisible to the outsider and then start working together to achieve the goal or they will work toward their personal ambitions. In other cases, where there is a defined process, practitioners are assigned to or voluntarily fill up the project roles. A

process is essential to the project. Whether the process is effective or not or the process is well defined or vaguely exists, process management is one of the main areas in project management.

IEEE's "Standard Glossary of Software Engineering Terminology" [32] defines the process as "a sequence of steps performed for a given purpose; for example the software development process." In the same standard, process management is defined as "the direction, control, and coordination of work performed to develop a product or a service. Example is quality assurance."

Two of the most widely recognized works mainly focus on improving project development and management processes. CMMI and earlier various CMMs are based on improving the maturity of organizations by improving their processes [8]. CMMI for development proposes specific and generic goals for each identified process area. As previously mentioned, PMBOK Guide identifies five project management processes groups: initiating, planning, executing, monitoring and controlling, and closing. Both CMMI and PMBOK Guide have a process-oriented approach to project management.

Endres and Rombach [33] present Humphrey's law as "mature processes and personal discipline enhance planning, increase productivity, and reduce errors." Many studies report that improved processes increase the likelihood of project success. As a result, inclusion of the process management into to 3PR framework is well-justified.

7.3.3 Product Management

According to PMBOK Guide, "a project is a temporary endeavor undertaken to create a unique product, service, or result." In the guide, the product is considered as the outcome of the project, which may be a product, service, or result. This view is also shared with Philips's definition of product: "the product is the project's final outcome" [12]. Products include software, firmware, documentation, reusable artifacts, training, and even services such as maintenance. The most important characteristic of the product is its quality. In every project, the stakeholders should come to a common understanding of what the product's quality should be. The earlier this common understanding is reached, the better it is. According to Blum, there are two views of quality: internal and external [34]. While internal quality is the developer's view of the software, external quality is the stakeholders' view of the software. Internal quality includes, but is not limited to, efficiency, testability, understandability, and modifiability. External quality includes usability, correctness, reliability, maintainability, integrity, etc. It is preferable to make these quality attributes as measurable as possible; however, this is not an easy task in every project. For example, a quality attribute such as usability may mean different things for the developers and the users. Thus, it is essential to define what usable means as early as possible in the project development. It is important to note that quality is not a

feature that can be included later in the product. It should be integral to the whole software development process.

7.3.4 Risk Management

As stated in PMBOK Guide, a project is undertaken to create a unique product, service, or result. This uniqueness is inherent and creates a certain amount of uncertainty in projects. This is also specifically addressed in Turner's theory of project management [35]. In this theory, the work in a project is non-routine and therefore risky. This is one of the inherent aspects of projects. Every project manager or project management team conducts risk management activities with different levels of rigor. The level of rigor varies from dedicated formal risk management procedures to ad hoc responses to risks.

Risk management has found its place in most well-established standards and guidelines such as PMBOK Guide, CMMI, Program Manager's Guide to Software Acquisition Best Practices [6], Guide to the Software Engineering Body of Knowledge (SWEBOK) 2004 Version [36], INCOSE's (International Council on Systems Engineering) Systems Engineering Handbook Version 3.1 [37], NASA Systems Engineering Handbook [38], and the Military Standard for Software Development and Documentation [39].

Boehm points out that in most software project disasters, the problems could have been avoided or reduced if the high-risk elements had been identified and resolved early on in the process [40]. Risk management has two primary steps: risk assessment and risk control. Risk assessment involves risk identification, risk analysis, and risk prioritization. Risk control involves risk management planning, risk resolution, and risk monitoring. Capers Jones identifies 60 software risk factors in his book [41]. This book is a good source of information for identification and resolution of risks in software projects.

Since risk management is an inherent aspect of project management, software project management framework includes risk management as a main area.

7.4 People Management

The people management main area includes seven project management areas. They are communication, teamwork, leadership, organizational commitment, project manager, stakeholder involvement, and staffing and hiring.

7.4.1 Communication

Communication can be generally described as "the exchange of ideas, opinions, and information through written or spoken words, symbols, or actions" [58]. A successful project requires constant and healthy communication between stakeholders. The importance of communication in projects is well established in the literature. Among all the project management areas listed in PMBOK Guide, communications management has the largest impact on project results [42]. Grinter expresses that good communication is vital to establish and maintain control over the software development process [43]. In the first four editions of PMBOK Guide, project communications management is listed as one of the project management knowledge areas. One notable change in PMBOK Guide fifth edition is dividing the project communications management into two separate project management knowledge areas: project communications management and project stakeholder management. Such change indicates that project communications management is still evolving. We contribute to this evolution by developing a simple model for project communications [66]. Note that 3PR framework has communication and stakeholder involvement as separate project management areas.

7.4.2 Teamwork

Teamwork may be defined as "the concept of people working together towards a common vision or a goal set as a team" [10]. Today, all medium- to large-scale software projects are developed by a team of people [60]. Moe, Dingsøyr, and Dybå state that the team is the basic work unit in innovative software organizations [59]. Achieving higher levels of teamwork is important for project success [10]. Furthermore, in one of our survey studies, we found out that one out of five software and information technology development projects is challenged in achieving teamwork [61]. Dingsøyr and Dybå argue that we need better models for team effectiveness in software development projects [60]. Dingsøyr and his colleagues hypothesize that team coordination, goal orientation, team cohesion, shared mental models, and team learning strongly influence software team performance [62]. Good coordination among team members is crucial for project success [63].

7.4.3 Leadership

Leadership may be defined as the ability to lead other people toward a shared goal or vision. Leaders inspire team members in a shared vision, and they successfully communicate this vision to other team members. This is important since half of the software or IT development projects are challenged in maintaining a well-defined

project scope [61]. Goal-oriented leaders set clear goals and milestones. They impact team performance and consequently project success [63]. Setting realistic goals and expectations is also an essential factor in managing a successful software project [64]. Our research shows that leadership effectiveness highly correlates with software project success [10]. In fact, our study indicates that leadership is the most important success factor in software projects [10]. In 3PR framework, leadership is not attributed to one person. Effective leadership is expected at every level. Note that in 3PR framework, the role of the project manager is considered a key role and treated separately as a project management area.

7.4.4 Organizational Commitment

Organizational commitment is the employee's psychological attachment to the organization and organizational goals [44]. In the project management context and in this framework, organizational commitment refers to the commitment to project organization and project goals. There is an important difference on how organizational commitment is viewed in this framework and other studies. In this framework, organizational commitment refers to commitment shown by all stakeholders including project team members. In most other studies, organizational commitment refers to the employee's commitment to organization.

7.4.5 Project Manager

The project manager position is a key role in project organizations. The project manager is mainly responsible for planning, directing, controlling, structuring, coordinating, and motivating in the project organization. In this study, project manager is considered as a role and authority as well as incorporating the necessary personal traits within the role. The role includes characteristics of both a good manager and a good leader. Our study shows that project manager is crucial to project success [10].

7.4.6 Stakeholder Involvement

Stakeholder involvement is the engagement and involvement of primary and secondary stakeholders during the project. This involvement includes, but is not limited to, planning, decision-making, development, testing, and implementation of the project. For a successful project outcome, stakeholder involvement is essential. After all, the project is undertaken to satisfy the needs of the stakeholders.

7.4.7 *Staffing and Hiring*

Staffing may be defined as "the practice of finding, evaluating, and establishing a working relationship with future colleagues on a project and detaching them from the project organization when they are no longer needed. Staffing involves finding people, who may be hired or already working for the company (organization)" [67]. In this definition, hiring is a part of staffing. However, in some organizations, hiring means employing project team from outside the organization, and staffing means employing project team members from the organization's various departments. To avoid confusion due to various definitions of terms, both terms are used in naming the area. In this framework, this area also includes the concept of placing the right people in the right role or position.

7.5 Process Management

The process management main area includes four project management areas. They are requirements management, project monitoring and control, project planning and estimation, and scope management.

7.5.1 *Requirements Management*

A definition of requirements management is "the management of all requirements received by or generated by the project, including both technical and non-technical requirements as well as those requirements levied on the project by the organization" [8]. In this framework, as the definition suggests, requirements management is the management of requirements and not the requirements development process. This is an important distinction. The requirements development process may be designed based on a specific software development life cycle model such as waterfall, spiral, agile, rapid prototyping, etc. The requirements management process itself is often independent of the development model.

7.5.2 *Project Monitoring and Control*

Project monitoring and control are actually two closely related project management areas combined into one area. Project monitoring is the process of keeping the project, project-related factors, and project metrics under continuous observation. Project control is the process of ensuring that project goes according to what is planned in the project plans and other documentations. In addition, the project

control process ensures that deviations from the plan are kept to a minimum and under control.

7.5.3 Project Planning and Estimation

CMMI 1.2 [8] defines the project planning as follows: "Project planning includes estimating the attributes of the work products and tasks, determining the resources needed, negotiating commitments, producing a schedule, and identifying and analyzing project risks. Iterating through these activities may be necessary to establish the project plan. The purpose of project planning is to establish and maintain plans that define project activities" [8].

Even though estimation is included in the previous definition, estimation exists in the title to make the term explicit and avoid any confusion. Project estimation includes creating and establishing estimates of project cost, schedule, and necessary resources using various methods, techniques, and tools.

7.5.4 Scope Management

In simple terms, scope management is the process of defining the scope of the project and keeping track of any changes in the scope. It also includes processes to limit the changes to the point that they are not disruptive to the success of the project. According to our study [61], scope management is the most challenging area in software and information technology projects.

7.6 Product Management

The product management main area includes two project management areas. They are configuration management and quality engineering.

7.6.1 Configuration Management

CMMI 1.2 [8] defines configuration management as: "A discipline applying technical and administrative direction and surveillance to (1) identify and document the functional and physical characteristics of a configuration item, (2) control changes to those characteristics, (3) record and report change processing and implementation status, and (4) verify compliance with specified requirements."

Sometimes the meanings of configuration management and scope management are mixed among software practitioners. However, the CMMI's definition of configuration management clarifies and stresses that configuration management is about managing the configuration items. These configuration items include intermediate and final project artifacts and products. Even though configuration management is a process itself, the focus of this area is the product. Therefore, configuration management is placed under the product main area to avoid confusion due to definition overload.

7.6.2 Quality Engineering

Quality engineering is another area placed under the product main area. Note that the term quality engineering is different from quality assurance. In many organizations, quality assurance is used to refer to procedures related to testing of the product. In others, it has a broader meaning. By using the term quality engineering, the framework widens the area and includes all the procedures and processes conducted to ensure products or services are designed and produced to meet or exceed customer requirements. Quality engineering involves all activities and commitment toward the development of a high-quality product to meet or increase the stakeholders' satisfaction.

7.7 Risk Management

There are two project management areas listed under the main area of risk management. They are risk assessment and risk control.

7.7.1 Risk Assessment

Risk assessment may be defined as "a process or a set of activities that involves identification, analysis, and prioritization of project risks." In some projects, risk assessment is conducted with quantitative and qualitative formal procedures and techniques, while in some others it is conducted as an ad hoc process. Whether it is formal or not, the quality of the project risk assessment also depends on the skills and experiences of the responsible project staff. According to Boehm, risk assessment involves risk identification, risk analysis, and risk prioritization [40].

7.7.2 Risk Control

Risk control may be defined as "the process of integrating findings from the risk assessment with technical, financial, policy, and non-technical concerns of stakeholders, to develop and select suitable risk control actions, and implementation of these actions. Risk control actions include implementation of policies, standards, procedures and physical changes" [45]. Risk control involves risk management planning, risk resolution, and risk monitoring [40]. To conduct an effective risk control, an effective risk assessment process has to be in place.

7.8 Validation of the 3PR Framework

We conducted an international survey study. Software project management and development practitioners from many different countries are invited to participate in this study. We used an online self-administered survey questionnaire to gather research data. There were two main goals of the study: (1) to validate the 3PR framework and to identify what is important in software project management and (2) to identify the project management challenges in software projects.

Before the main study, we conducted a pilot study with limited participation. The goal of the pilot study was to refine the survey questionnaire and identify the possible problems in the experimental design. Pilot studies are essential in survey studies. Especially if the research study is an exploratory study, testing the survey study design helps to eliminate many errors in design prior to the main study. The pilot study included an additional question that was left out later in the main study. In this question, we asked the survey participants how to improve the survey. The participants of the pilot study were randomly drawn from the pool of the sampling population of the main study. The pilot survey participants were not used again in the main study. The pilot study results led to improvements in the main study. The results and some of the improvements are listed as follows:

- Forty-four survey invitations were sent out. This population was randomly selected from the pool of the total sample population. There were 12 responses, yielding a response rate of 27.7%. This rate is almost the same as the response rate in the main study.
- One of the feedbacks indicated the necessity of a glossary section for the survey to eliminate possible misunderstandings. Therefore, a glossary section was added to the questionnaire.
- Two of the participants indicated the need for an explicit scale for a question in the paper version of the survey. Even though an explicit scale was not provided for this question, the participants were able to answer the question without difficulty. Later, a scale was added with the question.
- Two of the participants specifically indicated that all areas regarding the software project management were covered in the research.

- The questionnaire length was found to be reasonable.
- The participants found the questions understandable.
- The last question of the survey, inquiring about possible suggestions to improve the survey, was deleted in the main study.
- The analysis of the responses to a question regarding how important each main software project management area is as follows:

  People management = 39.16%
  Product management = 18.33%
  Process management = 25.00%
  Risk management = 17.50%

- The same ordering with similar ratings was found in the main study.
  The responses to a question regarding how important each software project management area were analyzed, and the ratings were ordered. The ordering of the ratings was significantly similar to the one gathered from the full-scale study.

The pilot study results showed that the survey instrument and the data collection procedures were found to be sufficient with the necessity of a few modifications and improvements.

In the main study, the survey questionnaire was sent out to more than 400 software development practitioners worldwide. We received 104 responses. The response rate was around 26%. Out of 104 responses, only 78 of the responses were acceptable. Some of the responses were eliminated due to lack of experience in software development or not completing all the necessary questions in the questionnaire.

The questionnaire had open- and closed-form questions. In some of the questions, we asked the survey participants to rate the importance of a specific project management area. In these questions, the rating was based on a seven-point Likert scale. The survey study was reported with all the details in [10].

In this study, we tried to reach a sample population that represents a wide range of software development roles. We asked the survey participants not just their current role in their last software project but all the roles they assumed during their career, because the participant's view on a particular concept is affected by the experiences they had during their software development career. The data related to survey participants is presented in Fig. 7.3. Note that the total number of responses is higher than the number of participants. The data in Fig. 7.3 included all the roles the survey participants assumed during their career. The data shows that most of the participants have management experience as a project manager or a project team leader. Furthermore, we had samples from a wide range of roles. The response "other" covers software development roles titled with other names.

In the survey study, we asked the participants to rate the importance of software project management areas. The response count for this question was 78 out of 104 respondents. Two more respondents filled out this question; however, they were eliminated due to lack of experience and not providing adequate background information. For each item in the question, the mean ratings were calculated. Then,

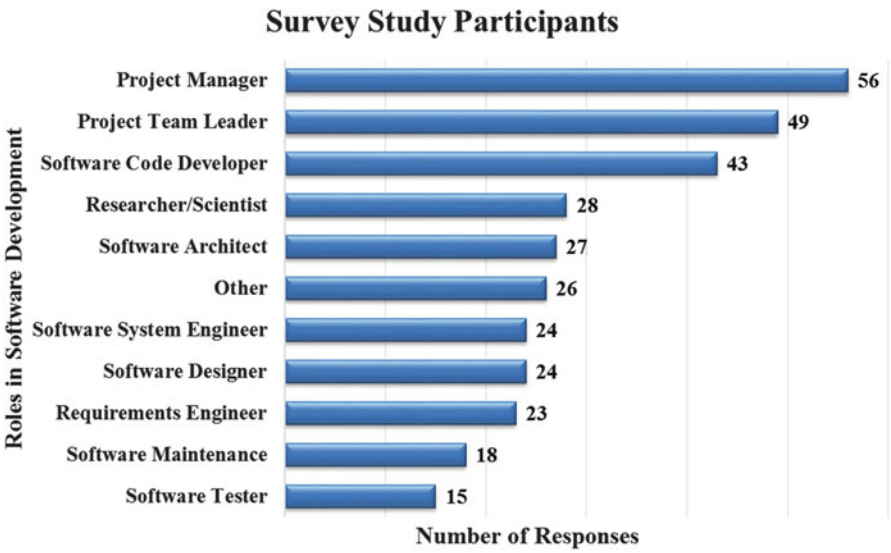


Fig. 7.3 Software development roles of the study participants

they are ordered from the highest to lowest. Prior to the study, we identified 17 areas. Table 7.1 includes 15 of these areas. The two other areas were "technical complexity" and "support activities (training, use of tools, etc.)." These two areas ranked last in terms of importance in software project management. Therefore, they are not included in the 3PR framework. One of the choices in the ratings was "no opinion." There were a significantly low number of "no opinion" responses. This means that almost none of the respondents had difficulty in associating the identified areas with software project management. This is attributed to the careful design of the questionnaire. In total, there were eight "no opinion" selections. These are, respectively, one in communication, one in scope management, one in staffing and hiring, two in quality engineering, and three in technical complexity.

There is a significant finding in the analysis of responses to these questions. The survey participants rated six of the software project management areas related to the people management area among the top seven of the ratings. This is also a confirmation to what will be found later in response to another research question. People management in software project management rated highest among the main areas. Process management related areas are found to be among the second highest ratings. The distinction between product- and risk-related areas is not as clear as people and process management related areas. However, note that the ratings are very close to each other between these project management areas. The importance ratings are presented in Tables 7.1 and 7.2.

In another question, the participants were asked to rate the importance of four main project management areas:

Table 7.1 Importance ratings of software project management areas

Software project management area	Means of ratings
Communication	5.69
Teamwork	5.41
Leadership	5.32
Requirements management	5.21
Organizational commitment	5.10
Project manager	5.09
Stakeholder involvement	5.05
Project monitoring and control	5.01
Project planning and estimation	4.99
Scope management	4.91
Risk control	4.86
Staffing and hiring	4.82
Configuration management	4.81
Risk assessment	4.72
Quality engineering	4.64

1. People (project manager, staffing and hiring, leadership, communication, teamwork, stakeholder involvement, organizational commitment)
2. Process (project planning and estimation, scope management, project monitoring and control, support activities, requirements management)
3. Product (quality engineering, technical complexity, and configuration management)
4. Risk (risk assessment, risk control)

We asked the survey participants to rate in such a way that the total rating of all four areas should add up to 100%. Figure 7.4 presents the research data gathered in response to this question. For each main area, we calculated the mean of the ratings.

The research data clearly shows that people management is the most important area in software project management. Process management is the second. Furthermore, the importance of product management and risk management is close.

In this research study, the order of the survey questions is carefully chosen to eliminate a possible bias due to the order of questions in the survey instrument. This careful design in the survey instrument enabled us to validate some of the responses. Note that the question related to the importance ratings of main areas is placed after the question related to importance ratings of each project management area. As mentioned earlier, the people management-related project management areas consistently ranked higher in terms of importance among all areas. Similar results are observed for process management related areas.

Based on the research results from the survey study, we concluded that the 3PR (people, process, product, and risk) framework for software project management is valid.

As stated earlier, one of the survey study goals was to identify the challenges related to project management in software projects. We asked the survey participants to report the challenging areas in their last project. Table 7.3 presents the

Table 7.2 Importance ratings of software project management areas – categorized under main project management areas

People management	**Means of ratings**
Communication	5.69
Teamwork	5.41
Leadership	5.32
Organizational commitment	5.10
Project manager	5.09
Stakeholder involvement	5.05
Staffing and hiring	4.82
Process management	**Means of ratings**
Requirements management	5.21
Project monitoring and control	5.01
Project planning and estimation	4.99
Scope management	4.91
Product management	**Means of ratings**
Configuration management	4.81
Quality engineering	4.64
Risk management	**Means of ratings**
Risk control	4.86
Risk assessment	4.72

Fig. 7.4 Importance of main software project management areas

survey results to this research question. There were also questions inquiring about the characteristics of their last projects. The study results indicated that almost half of the software projects are challenged in scope management and requirements management. Technical complexity was a challenge in only one quarter of software projects. Another notable result was that only 3% of the software projects were smooth. The detailed research results including multivariate analyses of the challenging areas are reported in a research paper [61].

Table 7.3 Challenging project management areas in software projects

Project management area	Response %	Response count
Scope management	52.6%	41
Requirements management	51.3%	40
Project planning and estimation	41.0%	32
Communication	38.5%	30
Staffing and hiring	33.3%	26
Project monitoring and control	28.2%	22
Risk control	26.9%	21
Technical complexity	26.9%	21
Stakeholder involvement	25.6%	20
Leadership	25.6%	20
Configuration management	25.6%	20
Organizational commitment	24.4%	19
Quality engineering	23.1%	18
Teamwork	21.8%	17
Risk assessment	19.2%	15
Project manager	14.1%	11
Other	10.3%	8
Support activities (training, tools, etc.)	9.0%	7
The last project was smooth in every way	2.6%	2

7.9 Conclusion

A framework for software project management titled 3PR (people, process, product, and risk) framework is presented in this chapter. The framework consists of four main software project management areas: people management, process management, product management, and risk management. Fifteen project management areas were identified and categorized under these main areas. People management area includes communication, teamwork, leadership, organizational commitment, project manager, stakeholder involvement, and staffing and hiring. Process management area consists of requirements management, project monitoring and control, project planning and estimation, and scope management. Product management area includes configuration management and quality engineering. Risk management area consists of risk assessment and risk control. These areas were identified with an extensive literature review and interviews with project management professionals. After the development of the framework, it was validated by a survey study with the participation of software practitioners. The 3PR framework and the survey study results are used in the development of a metric for software project management effectiveness [10].

3PR framework has many similarities with existing project management models and frameworks. Such similarity is natural since the current paradigm in project management discipline has not changed in recent years. However, there

are two distinctive characteristics of 3PR framework. First, people management area is considered quite important in 3PR framework. People management area has the most number of subareas in the framework. Furthermore, project manager has its own subarea in 3PR framework. This subarea is right on target since our research showed that the skills project manager possess and the leadership shown by project manager correlate high with software project success [10]. Choosing a good project manager is a good start for a successful journey in the project development. Second, risk management is considered a main project management area. While the importance of risk management is quite established, many project management models and frameworks are process oriented, and risk management is considered a process. In 3PR framework, risk management is considered a main project management area just like process management. Note that the "Program Manager's Guide to Software Acquisition Best Practices Version 2.31" starts with formal risk management. In the guide, risk management is considered vital to successful software acquisition. Our empirical research also showed the importance of risk management in software projects. Among all main project management areas in 3PR framework, risk management correlates highest with project success [10]. Furthermore, all main project management areas correlate high with project success [10]. Such findings strengthen the validity of the 3PR framework. As a result, 3PR framework is a valid alternative to existing frameworks in software project management.

Disclaimer and Acknowledgements The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any affiliated organization or government. This study was conducted as part of a doctoral research [10] on software project management.

References

1. Project Management Institute (PMI) (1996) A guide to the project management body of knowledge
2. Project Management Institute (PMI) (2000) A guide to the project management body of knowledge, 2000 edn
3. Project Management Institute (PMI) (2004) A guide to the project management body of knowledge, 3rd edn
4. Project Management Institute (PMI) (2008) A guide to the project management body of knowledge, 4th edn. ANSI/PMI 99–001-2008
5. Project Management Institute (PMI) (2013) A guide to the project management body of knowledge, 5th edn. ISBN: 978–1–935589-67-9
6. Software Program Manager's Network (SPMN) (1998) The program manager's guide to software acquisition best practices, version 2.31. Computers & Concepts Associates
7. CMMI Product Team, (2002) Capability maturity model integration, version 1.1, Software Engineering Institute, Carnegie Mellon University, March 2002
8. CMMI Product Team (2006) Capability maturity model integration, version 1.2. Software Engineering Institute, Carnegie Mellon University
9. CMMI Product Team (2010) CMMI for development, version 1.3. CMU/SEI-2010-TR-033. Software Engineering Institute, Carnegie Mellon University

10. Demir KA (2008) Measurement of software project management effectiveness. Doctoral Dissertation, Naval Postgraduate School, Monterey, California, USA
11. The Standish Group (2000) The standish group report: chaos
12. Philips D (2000) The software project manager's handbook, principles that work at work. IEEE Computer Society, Los Alamitos
13. El Emam K, Koru AG (2008) A replicated survey of IT software project failures. IEEE Softw 25(5):84–90
14. Wiegers KE (1996) Creating a software engineering culture. Dorset House Publishing, New York
15. Bach J (1995) Enough about process: what we need are heroes, IEEE Software, March
16. Curtis B, Hefley W E, Miller SA (1995) People capability maturity model, version 1.0. Software Engineering Institute, Carnegie Mellon University
17. Humphrey WS (1989) Managing the software process. Addison-Wesley, Reading
18. Humphrey WS (1996) Using a defined and measured personal software process. IEEE Softw 13(3):77–88
19. Humphrey WS (1997) Introduction to the personal software process. Addison-Wesley, Reading
20. Kulpa M (2007) Why Should I use the People CMM? Crosstalk - J Def Softw Eng:19–22
21. Brooks FP Jr (1975) The mythical man-month: essays on software engineering. Addison-Wesley, Reading
22. Bach J (1994) The immaturity of CMM. American Programmer, September
23. Curtis B, HefleyW E, Miller SA (2001) People capability maturity model, Version 2.0. Software Engineering Institute, Carnegie Mellon University
24. Humphrey WS (1995) A discipline for software engineering. Addison-Wesley., Reading
25. DeMarco T, Lister T (1987) Peopleware: productive projects and teams. Dorset House Publishing Company, New York
26. DeMarco T, Lister T (1999) Peopleware: productive projects and teams, 2nd edn. Dorset House Publishing Company, New York
27. Weinberg G (1994) Quality software management: volume 3 congruent action. Dorset House Publishing, New York
28. Cooke-Davies TJ (2004) Measurement of organizational maturity, Innovations – project management research, Chapter 13
29. Sackman H, Erikson WJ, Grant EE (1968) Exploratory experimental studies comparing online and offline programming performance. Commun ACM 11(1):3–11
30. Center for Software Engineering (CSE) at USC (1999) COCOMO II model definition manual. University of Southern California (USC), Los Angeles, USA
31. Hughes B, Cotterell M (2002) Software project management, 3rd edn. McGraw-Hill International (UK) Ltd, Berkshire
32. IEEE (1990) Standards coordinating committee of the computer society of the IEEE. IEEE Standard Glossary of Software Engineering Terminology. IEEE, New York
33. Endres A, Rombach D (2003) A handbook of software and systems engineering empirical observations, laws and theories. Pearson Education, Essex, England
34. Blum B (1992) Software engineering, a holistic view. Oxford University Press, New York
35. Turner JR (2006) Towards a theory of project management: the nature of the project. Int J Proj Manag 24:1–3. 93-95, 187-189
36. IEEE (2004) Guide to the software engineering body of knowledge (SWEBOK), 2004 Version
37. International Council on Systems Engineering (INCOSE) (2003) Guide to the systems engineering body of knowledge
38. National Aeronautics and Space Administration (NASA) (2007) Systems engineering handbook
39. Department of Defense (DoD) (1995) MIL-STD-498 military standard software development and documentation
40. Boehm BW (1991) Software risk management: principles and practices. IEEE Softw 8(1):32–41
41. Jones C (1994) Assessment and control of software risks. Prentice-Hall, Englewood Cliffs
42. Muller R (2003) Determinants for external communications of IT project managers. Int J Proj Manag 21:345–354

43. Grinter RE (1996) Understanding dependencies: a study of the coordination challenges in software development. Doctoral dissertation, University of California, Irvine, USA
44. Brown BB (2003) Employees' organizational commitment and their perception of supervisors' relations-oriented and task-oriented leadership behaviors. Doctoral dissertation, Virginia Polytechnic Institute and State University, Falls Church, Virginia, USA
45. LesRisk (2008) Definition of risk control. http://www.lesrisk.com/glossary.htm. Accessed 25 Sep 2016
46. Futrell RT, Shafer DF Safer LI (2002) Quality software project management. Prentice Hall
47. Forsberg K, Mooz H, Cotterman H (2005) Visualizing project management. Wiley, Hoboken
48. The Standish Group (1995) The standish group report: chaos, West Yarmouth, MA
49. Slevin DP, Cleland DI, Pinto JK (2002) The frontiers of project management research. Project Management Institute, Pennsylvania
50. Yourdon E (2003) Death March, 2nd edn. Pearson Education, Inc. Publishing as Prentice Hall Professional Technical Reference, Upper Saddle River
51. Jones C (2004) Software project management practices: failure versus success. Crosstalk - J Def Softw Eng 17(10):5–9
52. Cicibas H, Unal O, Demir KA (2010) A comparison of project management software tools (PMST). In: Proceedings of the Software Engineering Research and Practice (SERP 2010), pp 560–565, July 12–15, 2010, Las Vegas, Nevada, USA
53. Robertson S, Robertson J (2005) Requirements-led project management. Pearson Education, Inc., Boston
54. Defense Science Board (DSB) (2000) Report of the defense science board task force on defense software
55. Boehm BW (1981) Software engineering economics. Prentice-Hall, Upper Saddle River
56. Boehm BW, Madachy R, Steece B (2000) Software cost estimation with COCOMO II, Prentice Hall PTR
57. Thomsett R (1995) Project pathology: a study of project failures. American Programmer, July, 8–16
58. Chowhan SS, Shekhwat N (2015) Business and management. Lulu Online Publishing
59. Moe NB, Dingsøyr T, Dybå T (2009) Overcoming barriers to self-management in software teams. IEEE Softw 26(6):20–26
60. Dingsøyr T Dybå T (2012) Team effectiveness in software development. 5th international workshop on Cooperative and Human Aspects of Software Engineering, (CHASE 2012)
61. Demir KA (2009) A survey on challenges of software project management. Proc. Software Engineering Research and Practice (SERP 2009), pp 579–585, Las Vegas, NV, USA, 13–16 July 2009
62. Dingsoyr T, Faegri TE, Dyba T, Haugset B, Lindsjorn Y (2016) Team performance in software development: research results versus agile principles. IEEE Softw 33(4):106–110. doi:10.1109/MS.2016.100
63. Hoegl M, Gemuenden HG (2001) Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence. Organ Sci 12:435–449. Jul-Aug 2001
64. Reel JS (1999) Critical success factors in software projects. IEEE Softw 16(3):18–23
65. Curtis B, Hefley B, Miller S (2009) People capability maturity model (P-CMM) version 2.0, Second Edition, Software Engineering Institute, Carnegie Mellon University, July 2009, CMU/SEI-2009-TR-003
66. Demir KA (2010) A simple framework for project communications. Proc. Software Engineering Research and Practice (SERP 2010), pp. 452–458, Las Vegas, NV, USA, 12–15 July 2010
67. Talloo TJ (2007) Business organization and management. Tata McGraw-Hill Education, May 1, 2007

Chapter 8
CrowdSWD: A Novel Framework for Crowdsourcing Software Development Inspired by the Concept of Biological Metaphor

Tarek A. Ali, Eman S. Nasr, and Mervat H. Gheith

8.1 Introduction

From distributed computing (DC) point of view (see Fig. 8.1), the symmetric multiprocessing architecture means multiple processors sharing the same memory. The parallel cluster architecture means multiple machines; each may consist of several CPUs and memory. The grid computing architecture goes further by connecting both architectures, symmetric and cluster machines, from different computing sites. It aims to reach common goals by using collections of machine resources from multiple locations [1]. The cloud computing architecture integrates and aggregates a lot of independent machines and software to virtualize and provision them by utilizing virtualization techniques. It aims to reach coherence and economies by using large groups of remote servers [2]. This computing architecture has enabled us to combine and harness intelligence, knowledge, and life experiences of people to do tasks or produce information that is hard for individual users or computers to achieve alone [3].

Mainly, there are two types of computing elements (CEs), which can be used in DC; they are either machine-based computing element (MBCE) or human-based computing element (HBCE). MBCE is a standard CE and component of stream processing systems, e.g., in [4]. It executes a set of operations in a fully automated manner on its input stream. It is an unprecedented power to transmit information over large distances, to store and manipulate data for a long time, and to quickly perform well-defined formal computations. HBCE is a human ability for

T.A. Ali (✉) • M.H. Gheith
Department of Computer Science, Institute of Statistical Studies and Research, Cairo University, Giza, Egypt
e-mail: tarekmmmmt@pg.cu.edu.eg; mgheith@cu.edu.eg

E.S. Nasr
Independent Researcher, Cairo, Egypt

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_8

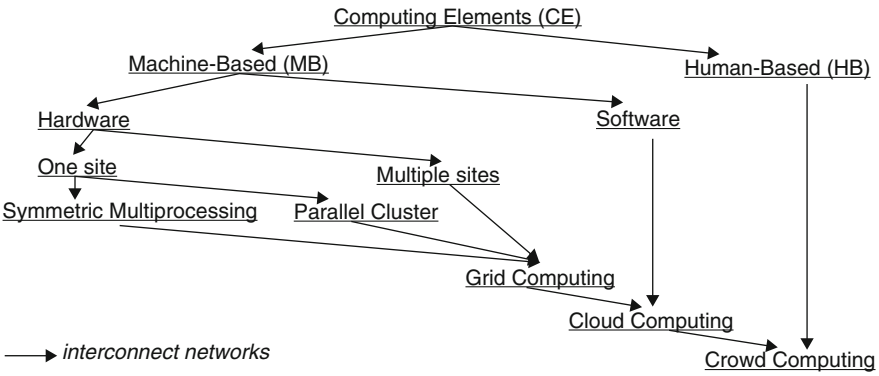


Fig. 8.1 Distributed computing and computing architectures

computation to solve problems that are trivial for humans but complex for machines. It depends on competencies, knowledge, and skills with networks of social relationships and understanding of social context [5]. However, hybrid-computing element (hybrid-CE) comprises both MBCE and HBCE in a hybrid class. The expression "hybrid-CE" has been inspired from the biological systems (see Fig. 8.2). Such systems have the ability to manage themselves according to an administrator's goals. Moreover, this hybridity can be deployed and utilized as a collective on demand based on a different quality, cost, time, and incentive models. That means the hybridity supports human task where the HBCE do the originative work and the MBCE does the management. Such as stream processing systems in [4], hybrid-CE consumes data items and control flow signals through input ports, implements an application-specific requirement, and emits the processed data items through an output port. Hence, the data and information are exchanged by passing messages between the ports using the available communication links. Here, the ports might be an MBCE or HBCE. Moreover, it might be a computing node in the computing architecture. Hence, hybrid-CE is ubiquitous and can, for instance, be of biological, technical, or social nature [6, 7]. There are several frameworks that can then be used for mapping hybrid-CEs to formal definitions in most of the IS textbook literature [8, 9]. For example, hybrid-CE may be implemented through social machines [10] or through the API of a remote crowdsourcing platform, e.g., Amazon's Mechanical Turk, CrowdFlower, TopCoder, or others.

Starting from the several conceptual and empirical studies about tasks management, Fig. 8.3 integrates some of them by showing the representation of the visual field, which greatly simplifies the development and management complexity of crowdsourcing software. For example, socially intelligent computing (SIC) extends traditional management processes as shown in Fig. 8.3a. It aims to understand the ways in which systems of human intelligence across the globe and social platforms can work together as efficiently as a giant machine [3, 11]. It supports self-* properties such as self-organizing, self-maintenance, self-control, self-evaluating, self-awareness, or even self-management [12]. Self-* might lead to what we call *-

Fig. 8.2 A beehive as an example of complex biological systems

family properties such as maintainability, flexibility, testability, usability, portability, reusability, and interoperability, which are used in the quantitative evaluation of software quality [13]. That is going to happen with the crowdsourcing software development and management also. No matter how good crowd members are at anything, the development or management is not finished until they interact with others because different versions of system development life cycle (SDLC) prescribe different numbers of phases or building blocks and different degrees of interaction between them [14]. For example, as shown in Fig. 8.3b, there are three versions of SDLC: solid arrows mean the forward-only version; solid and dashed arrows mean a version allowing backtracking; and solid, dashed, and dotted double arrows mean a clique, crowd, or even social network users (SNUs) version. Because the transition between any two activities is allowed, bearing this in mind, we drive a social machine [15] based on both SIC and SDLC. This machine supports real-time collaborative software development, allowing multiple users to work on the same collection of files or project with each other. A good scenario runs as shown in Fig. 8.3c. Adam wants to develop and manage his own software module using a social machine, a typical test development procedure might run as follows:

1. Adam asks Bob to write some tests.
2. Bob writes a test for the first method.
3. Bob runs the test and it passes.
4. Bob then creates tests for the rest of the methods.
5. Bob finds that many of the tests fail.
6. Bob messages Adam to say he has completed the tests, and many of them fail.

At this point, Adam merges in Bob's changes, to be able to modify the main code to pass the tests. That means Adam initiates the business model at a high level, and then the development and management processes are distributed among different computing elements, whatever human-based or machine-based computing elements. Therefore, the research question is how to empower social machines and other related works? Alternatively, how to empower Adam as a crowdsourcer. A crowdsourcer is any entity (company or individual) that has the means to carry out a certain initiative. Crowdsourcer is derived from the word "crowdsourcing." Crowdsourcing is defined to be a business practice that outsources an activity to

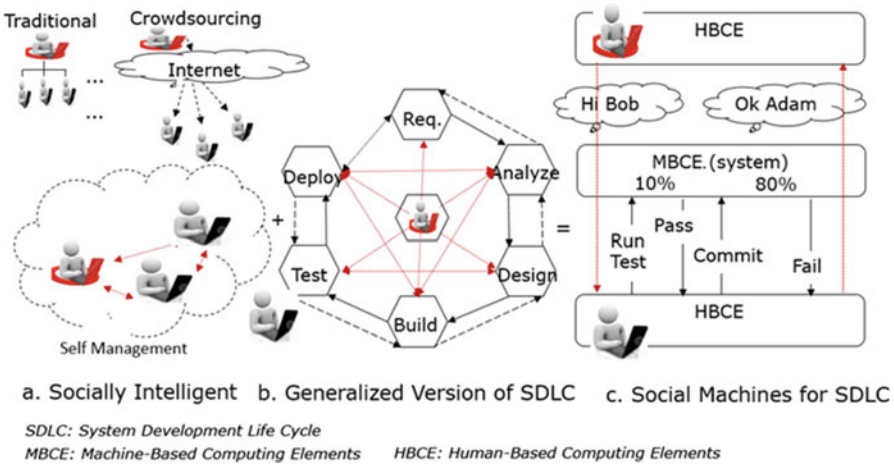


Fig. 8.3 Different modes of crowdsourcing software development and management

the crowd. The crowd is a community whose characteristics will be determined by the requirements of the crowdsourced initiative [16].

Reframing the problem, there are indicatives of three core sets of problems we face – those concerned with creating tasks for HBCE, making these tasks clear and simple, and structuring or managing these tasks developing for the crowd. To address this, we found six different aspects or building blocks of biological metaphor, e.g., honeycomb, including social, experience, motivation, limitation, management, development activity, and task blocks. Each block was evaluated, and an initial solution was selected based on the beehive collaboration model, which contains essential features of the user-user or bee-bee and user-task or bee-wax interactions.

The rest of this chapter is structured as follows. Related work is discussed in Sect. 8.2. High-level and low-level designs of a special purpose framework are presented in Sects. 8.3 and 8.4, respectively. This framework works as an instance of system development life cycle, which can be used to solve crowdsourcing software problems. Section 8.5 presents the design of 34 heterogeneous computing elements that can be used in crowdsourcing software development. An experimental study is discussed in Sect. 8.6. The results from the experiment and recommendations are given in Sect. 8.7 before we conclude in Sect. 8.8.

8.2 Related Work

SIC is a new emerging field that refers to the recent efforts of crowd computing (CC) to understand the ways in which systems of human intelligence across the globe and social networks can work together as efficiently as a giant machine. This

machine aims to share ideas or to solve a problem that cannot be easily done only by human intelligence tasks without technological platforms or vice versa [17–21]. These systems result in new behaviors that occur as a result of the complex interaction between humans and computers and will prove useful as intermediaries between the crowd and technological forms used by crowds [22–25]. Typically, these systems are based on social platforms through specific components, such as a graph API [26], authentication [27], social plugin [28], and open graph protocol [29]. Hence, the core of those systems is sets of hybrid-CEs.

Mainly, there are four categories of hybrid-CE over SIC or unified collective intelligence. Each of which relies on a machine (M) and human (H). The following examples demonstrate the four categories in developing complex applications. They figure out how machine guides human and vice versa:

1. HH: A human-based genetic algorithm [30] which uses both HBCE in terms of selection and types of human-based innovation. Thus, all HBCEs of a typical genetic algorithm are outsourced to humans, e.g., integrating crowds with a genetic algorithm to study creativity. Collaborative filtering in social search applications [31] accepts contributions from the crowd and attempts to use human evaluation to select the fittest contributions that get to the top of the list.
2. HM: Such as the gamification [32, 33] which is the application of typical elements of game playing. In computerized tests, M generates a problem and search for HBCE, e.g., CAPTCHA [34] tells human users from computer programs by presenting a problem that is supposedly easy for a human and difficult for a computer. This refers to human swarming or social swarming [35] in a real-time closed-loop systems, working around groups of networked users melded after biological swarms, and enabling human participants to behave as unified collective intelligence [35].
3. MH: Mainly, this category is based on human-computer interaction (HCI) that enables the actor to create an abstract drawing only by selecting his/her favorite images. Therefore, human only performs the selection, which is easier for humans, and software performs the innovative role.
4. MM: Such as domain knowledge-based and automatic workflow generation [36].

There are a number of approaches, frameworks, methods, models, SDLC, and software project management for DC, which could be adapted or further developed to investigate hybrid-CE. For example:

- Barowy et al. [37] claimed that "AUTOMAN" is the first fully automatic "crowdprogramming" SDLC. It integrates HBCE into MBCE (standard programming language) as ordinary function calls, which can be intermixed freely with traditional functions. It focuses on the programming logic and specifies a confidence level for the overall computation and a budget.
- Dwarakanath et al. [38] developed "CrowdBuild" SDLC that the implementation of the code can be done using crowdsourcing. Here, SDLC consists of four steps, which are designed based on a hierarchical component architecture

(HCA): specification of contracts for components of the HCA, implementation of the contract by the crowd, and then the automated verification and integration are done.

- Automatic personalized interface generation, that is, a feasible and scalable solution to perceptual and cognitive ability challenges [39]. Amsterdamer et al. [40] developed a natural language framework to crowd mining interface. This interface combines searching knowledge bases for general data with mining the crowd for individual and unrecorded data. It interacts with the end user during the development process to resolve uncertainties and complete missing data.
- Bernstein et al. [41] discussed the find-fix-verify crowd programming framework, which splits tasks into a series of generation and reviews stages. Those stages are achieved from components and algorithm development to project concept formations.
- Wenjun et al. [42] presented another interesting creativity in development processes that is called "AppStori." It is essentially an online IOS application market where developers can directly deliver their products with creative design to smartphone users. These developers are motivated to contribute innovative design and attract more user downloads by the micro-payment mechanism of the App Store. Within less than 4 years, it becomes a huge mobile application ecosystem with over 150,000 active developers and over 700,000 IOS applications. It is also the largest online software distribution channel for IOS applications.
- TopCoder model [43] analyzes the overall development process including activities performed at each step and deliverables produced during the process. Mainly, it follows a two-step approach. First, it examines the competition rules used in software crowdsourcing and their implications to activities that will be performed, and quality of software will be produced. Then, it identifies the people involved, the time they are involved, and the kind of activities involved including evaluation. *TopCoder and AppStori processes identified the min-max nature among participants as an important design element in software crowdsourcing for software quality and creativity. Although in a min-max game, one party tries to maximize the finding of bugs in a set of artifacts, and the other parties try to minimize the potential bugs in the same artifact; software crowdsourcing can still be a collaborative and win-win process for all parties. By using this approach, many aspects of software development can be crowdsourced with the crowd who can contribute their creativity to each aspect.*
- Kulkarni et al. [44] explored "Turkomatic" that uses a general-purpose divide-and-conquer algorithm to solve arbitrary natural language requests posed by end users. It has a new interface to microwork platforms that use crowd workers to help in planning workflows for complex tasks. The interface includes a novel real-time visual workflow editor that enables crowdsourcer to observe and edit workflows while the tasks are being completed.

The verification of work and the division of labor among members of the crowd can be handled automatically (MBCE) by Turkomatic, which substantially simplifies the process of using human computation systems (HBCE). These characteristics enable a novel means of interaction with crowds of online workers to support successful execution of complex work.

- Some researchers [36] proposed a language called "CrowdWON." It is able to represent the workflow of most well-known existing applications, extend previous modeling frameworks, and assist in the future generation of crowdsourcing platforms. CrowdWON is a new graphical framework used to describe and monitor crowd processes. It allows for the formal definition of adaptive workflows, which depend on the skills of the crowd workers and/or process deadlines. Moreover, it allows expressing constraints on workers based on previous individual contributions.
- Franklin et al. [45] claimed that some SQL queries cannot be answered by MBCE only. Processing such queries requires HBCE for providing information that is missing from the database, for performing computationally difficult functions, and for matching, ranking, or aggregating results based on fuzzy criteria. Therefore, they explained the presented initial solutions through a simple SQL schema and query extensions that enable the integration of HBCE and MBCE. They designed a new HBCE, e.g., crowdsourced query operators and plan generation techniques that combine both crowdsourced (HBCE) and traditional query operators (MBCE). They described methods for automatically generating effective user interfaces (hybrid-CE) for crowdsourced tasks. They presented a general architecture of "CrowdDB" that works as an application issues requests using "CrowdSQL" or a moderate extension of standard SQL.
- Ahmad et al. [46] presented a social computing stack "Jabberwocky" that consists of three components, viz.:

  (1) Dormouse, a human and machine resource management system. It is designed to enable cross platform programming languages for social computation. It trades both machines and people as first-class citizens. It is based on allowing the natural parallelization and control flows for a broad range of data-intensive applications. And finally and importantly, it includes notions of real identity, heterogeneity, and social structure.

  (2) ManReduce, a parallel programming framework for human and machine computation.

  (3) Dog, a high-level programming language on top of ManReduce.
- Minder et al. [47] presented a "Crowdlang." It is a concept of an executable, model-based programming language and a general purpose framework for accomplishing more sophisticated problems through managing dependencies between CEs. Their approach is inspired by coordination theory and an analysis of emergent collective intelligence.

There are many researchers who developed a self-conscious design process. They formulated a process theory of software design practice. This theory explains how collocated software development teams in organizations create complex software

systems. It posits that an independent agent (design team) creates a software system by alternating between three activities: organizing their perceptions about the context, mutually refining their understandings of the context and design space, and manifesting their understanding of the design space in a technological artifact [14]. It is based on creating a formal model of the mental pictures through the set theory and then solves problems using a divide-and-conquer strategy. The bootstrapping social machines [15] are used in the creation of crowdsourcing software. It is based on crowdsourcing projects such as TopCoder. Moreover, several frameworks that are aiming to support such new collaboration models are being developed (such as socially enhanced computing [48, 49]). These new frameworks are intended to support greater task complexity, more intelligent task division for a complex organizational system, and managerial structures for virtual teams. For example:

- Ham et al. [50] weave visual analysis of social relationships into software development, leading to the notion of relationship-aware software [51]. Blending computational (MBCE) and social elements (HBCE) into software appears as a promising framework for unifying both computational and social processes.
- Some researches [10] aim to investigate efforts related to this topic and build a preliminary classification scheme to structure the science of social machines. These researches can collaborate to the process of providing a more common and coherent conceptual basis for understanding social machines as a paradigm. For example, they characterized the "social machines" paradigm as a result of the convergence of three different visions: (1) social software, (2) people as computational units, and (3) software as social entities.

Through crowdsourcing software developments and social machine, there are some honeycomb frameworks, which could be used in crowdsourcing software developments such as social software building blocks and user experience building blocks. Social software building blocks that are used to develop more interactive and innovative applications allowed users to interact with each other and have a primary role as producers of content [52]. User experience building block is used to help people understand the need to define user priorities by asking many questions such as is the application useful to the individual user and the specific task? And is the application desirable for the individual user and the specific task?

Finally, there are many researchers [53] who targeted the goal of the future crowd workplace in which we would want our children to participate. Hence, they framed the major challenges that stand in the way of this goal. Their framework integrates the challenges posed by managing shared resources (such as assigning workers to appropriate tasks), managing producer-consumer relationships (such as decomposing tasks and assembling them into a workflow), and crowd-specific factors (such as motivation, rewards, and quality assurance). Many of its CEs combine insights from organizational behavior and DC.

8.3 CrowdSWD: High-Level Design

The crowdsourcing software is a broad term that describes large-scale distributed systems that comprise of many heterogeneous computing elements, each of which may have its own individual characteristics, objectives, and actions. Our society increasingly depends on such systems, in which collections of heterogeneous computing elements are tightly entangled with human and social structures to plan collective intelligence. The premise of the research is that existing frameworks for crowdsourcing software development are not powerful enough to cover large classes of aspects-relevant problems. Reframing the problem, there are indicatives of three core sets of problems we face – those concerned with creating tasks for HBCE, making these tasks clear and simple, and structuring or managing these tasks developing for the crowd. To address this, we found six different aspects or building blocks of biological metaphor, e.g., honeycomb, including social, experience, motivation, limitation, management, development activity, and task blocks. Each block was evaluated, and an initial solution was selected based on the beehive collaboration model, which contains essential features of the user-user or bee-bee and user-task or bee-wax interactions. To address this, we explored one instance of SDLC, which can be used to solve those problems. It follows a three-layered approach. It comprises building blocks, self-organization, and hybrid-CE layer. Employing an analogy from biology where the operator relating the crowd and the parameter to be social phenomena is assumed to belong to semi-algebraic sets (see Fig. 8.4). Building block layer plays as contextual information for handling the diversity of changes in and conditions of crowd's surrounding environment. Consequently, self-organization layer is used to acquire, analyze, model, manage, and then adapt this contextual information for the crowd. Finally, hybrid-CE layer is used to cope with different scenarios and options and introduces an extra level of skill, as crowd's needs at runtime decisions, depending on varying context conditions.

Theoretically, CrowdSWD aims to empower the crowds who do not necessarily have time or experience in software development, to create a small part of crowdsourcer's software, and to address crowdsourcer's needs. It is an interdisciplinary field of ubiquitous computing, tangible and embodied interaction, and online communities. It emerges as an approach to empower the crowds, transforming their role from a passive audience to active creators of their technological habitat. CrowdSWD is an open-source version control and collaboration framework for crowdsourcing software development and management. It works as crowd-centered design (CCD) method [3] and adaptation engine (AE). CCD is an iterative step-by-step process for developing a CE with the participation of the SNUs [54]. It defines variations, which determine and generate the best hybrid-CE by using workers' profiles in an independent manner. An independent variation of workers' profiles could be the evaluation of different building blocks by using forced agreement or guessing in a parallelized manner and then to automatically generate the best hybrid-CE based on the results of this evaluation.

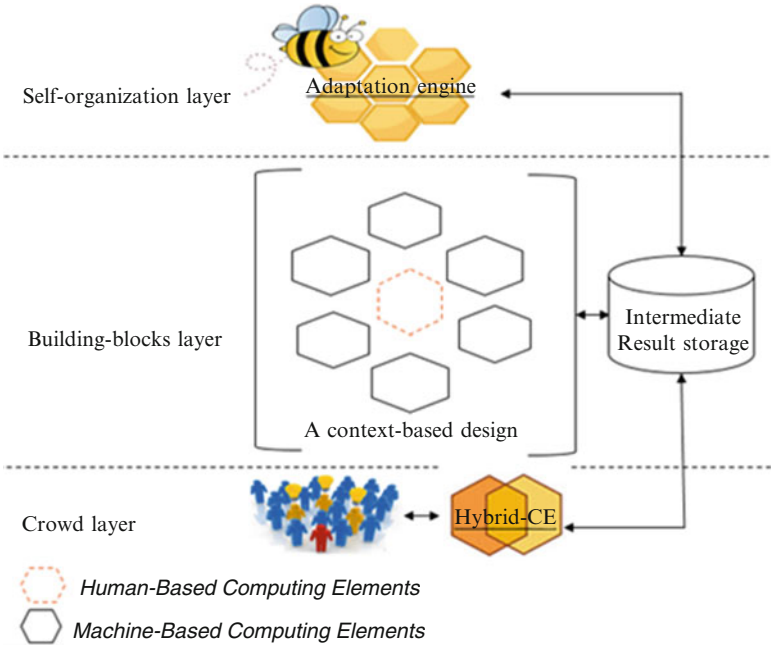


Fig. 8.4 A proposed framework for crowdsourcing software

Technically, CrowdSWD allows SNUs to develop, change, adapt, and improve software from its public repositories. SNUs can follow each other, rate each other's work, receive updates about specific projects, and communicate publicly or privately. The next sections present a high-level design for a special purpose framework "CrowdSWD," which can be used to solve crowdsourcing software problems, following a three-layered approach.

8.3.1 Building Block Layer: Context-Based Design

Building blocks appeal to our notion of problem decomposition and the assembly of solutions from sub-solutions. From SDLC point of view, SDLC is a set of building blocks. These blocks provide an organization with hybrid-CEs needed for operations and management [9, 55, 56]. In the crowdsourcing software development, most blocks are open, i.e., they interact with their crowd via interfaces. For example, the Rational Unified Process (RUP) [57] is an iterative software development process framework. RUP is based on a set of building blocks or content elements, describing what is to be produced, the necessary skills required, and the step-by-step explanation describing how specific development goals are to be

achieved. Today, one of the reasons behind the limited use of planned collective intelligence is the lack of MBCE for doing so and the little understanding of dependencies among different building blocks in problem-solving. These blocks needed to efficiently formulate and solve crowdsourcing software problems. Employing an analogy from biology, we call these building blocks the "genes" of collective intelligence systems, the conditions under which each gene is useful, and the possibilities for combining and recombining these genes to harness crowds effectively. As shown in Fig. 8.4, we have six outer blocks. Each presents one aspect of the crowdsourcing software. Each aspect has six facets. Each facet can be represented via MBCE or HBCE. Each HBCE goes to the inner block, the seventh block, or the center block.

8.3.2 Self-Organization Layer: Adaptation Engine (AE)

SIC is aiming to support self-* properties such as self-organizing, self-maintenance, self-control, self-evaluating, self-awareness, or even self-management [12]. Self-* might lead to what we call *-family properties such as maintainability, flexibility, testability, usability, portability, reusability, and interoperability, which are used in the quantitative evaluation of software quality [13]. It is important to mention, that especially generated hybrid-CE by an AE can use a rule-based engine and a huge set of MBCEs such as statistical models e.g., predicting the best hybrid-CE in an iterative collaboration by predicting a dependency, associations, combination, decision, and generation rules on the number of iterations. For example, during each iteration in the development process toward a solution, all the development points are updated using neighboring values in the mesh network. Therefore, hybrid-CE over rule-based engine means that AE requires the selection of a set of previous building blocks (e.g., web contents, exercises, a color, a path to follow, etc.), as well as the selection of these blocks, can be made with a set of rules. These rules can be usually divided from a logical perspective into smaller parts of hybrid-CEs that can be repeated in other composed rules in the application, as well as in other external applications. In an analogy between the atomic rules and the blocks in a honeycomb model: there is no need to code several times the same atomic rules, which implies the simplification for the creation of larger rules, reusability, interoperability, or easier maintainability at the low level of hybridity. Moreover, these rules perform adaptation by changing themselves during the time to accommodate resource variability. At the same time, AE has to choose one or more specific resources in a specific moment (e.g., a complete web page to present to a user). The selection process might depend on many different facets such as the user preferences and other resources.

8.3.3 Hybrid-CE Layer: Engineering Crowdsourced Stream CEs

A similar design of hybrid-CE is a CAPTCHA (completely automated public Turing test to tell computers and humans apart) [34]. It blends computational MBCE and HBCE into software module as shown in Fig. 8.5. It was found that the quality of the extracted information from images via human as HBCE is better than extracted information via optical character recognition (OCR) as MBCE.

In order to better understand what a hybrid-CE is, we follow a crowdsourced stream processing system [4] and a distributed stream computing platform [58]. A system can be examined from two main perspectives: (1) a behavioral or teleological aspect, where the system's behavior and goals are examined and (2) A structural aspect, where the system's structure, architecture, and operations are assessed. These aspects are supported by notions that include environment, goal, function, CE, and relationships between them, as illustrated by the system model of Fig. 8.6. Collaboration environment is anything that supports the hybrid-CE's boundaries. It influences the hybrid-CE as well as the hybrid-CE influences it. The collaborative environment represents a context-based virtual rendering of a user environment.

A crowdsourced stream CE is a hybrid-CE that incorporates crowdsourced tasks (HBCE) in the distributed computing architecture. This can be seen as enabling HBCE to be applied on a sample of large-scale computing at high speed, or equivalently, enabling stream computing to employ human intelligence. Engineering a hybrid-CE requires the combination of HBCE and MBCE. From a general distributed computing perspective, this means taking into account inherited as well as emerging aspects from both these CEs.

Fig. 8.5 A CAPTCHA as a typical example of hybrid-CE

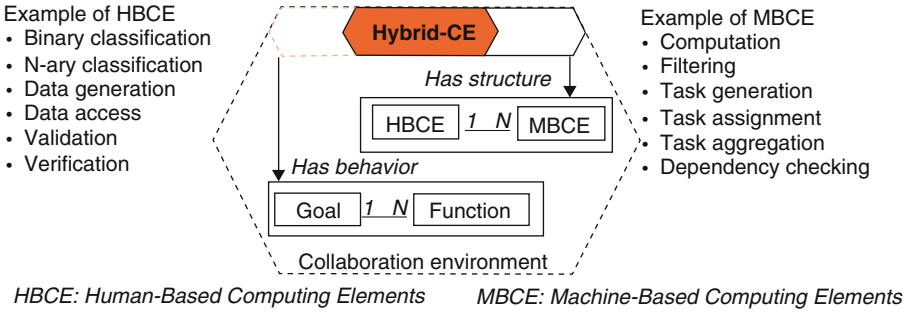


Fig. 8.6 Simplified meta-model of hybrid-CE

8.4 CrowdSWD: Low-Level Design

SNUs such as Bob and Adam use important terms such as fork, pull request, and merge. A fork is simply a repository that has been copied from one SNU's account to another SNU's account. Forks allow a SNU to develop and make modifications without affecting the original code. If the SNU would like to share the development and modifications, she/he can send a pull request to the owner of the original repository. If after reviewing the modifications, the original owner would like to pull the modifications into the repository, she/he can accept the modifications and merge them with the original repository. The aim of CrowdSWD is so intuitive to be used as version control and collaboration tool, nonprogrammers or even children will begin to use CrowdSWD to work on document-based and sketching projects. To achieve that aim, the next sections present a low-level design (see Fig. 8.7) for each layer as follows.

8.4.1 A Context-Based Design

As shown in Fig. 8.8, we have six outer blocks. Each presents one aspect of the crowdsourcing software. Each aspect has six facets. Each facet can be represented

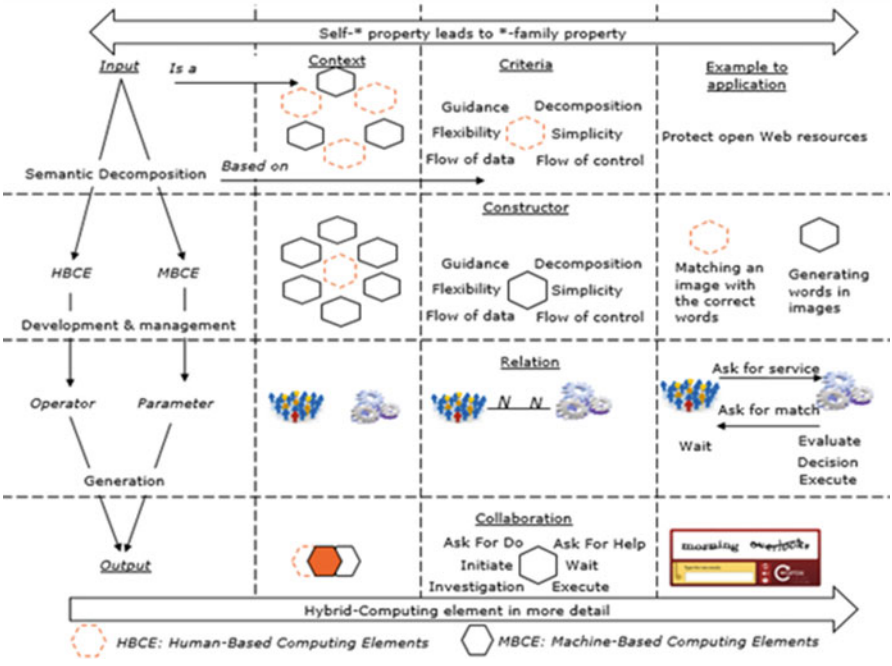


Fig. 8.7 CrowdSWD: low-level design

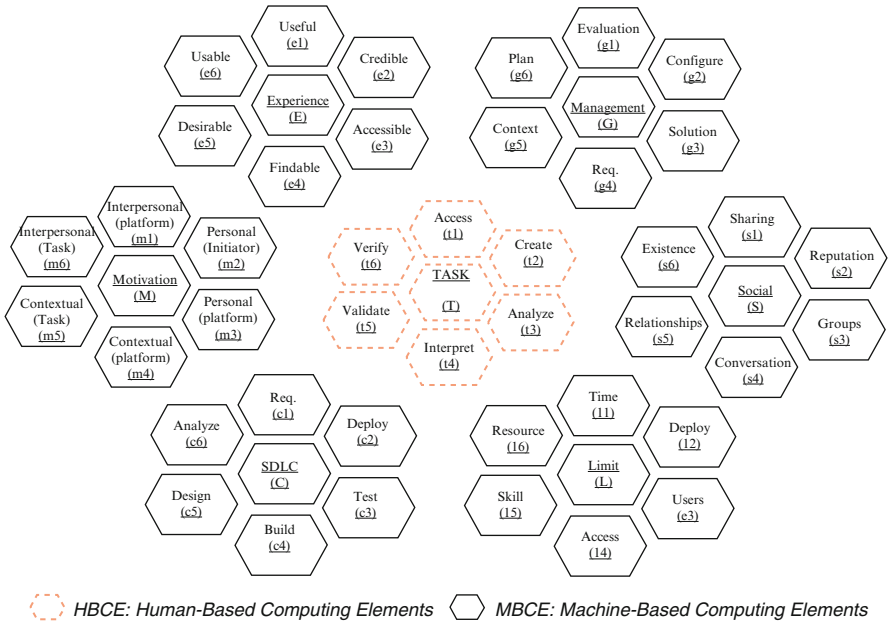


Fig. 8.8 A context-based design

via an MBCE or an HBCE. Each HBCE goes to the inner block, the seventh, or the center block. Now and return to Adam and his social machine, for example. Adam wants to know whether Bob is online. It is really easy; it is an existence facet, s small six of social block or social aspect or S capital; it is an MBCE of knowing who is online, such as any social platform, whether Facebook or others. Adam asks Bob to write some tests for a module. This means that Bob has a verification, t6, and validation, t5, skills during task completion. These skills are task block, task aspect, capital T, and the inner or center block. Relating to the other blocks as follows:

- Social block (S) is based on representing the six facets of SNU, including their social ties, interests, capabilities, activity history, and topical affinities. In [59], they defined, designed, and implemented a comprehensive model able to cater for all the aspects relevant for the applications involving social networks, human computation, and gaming activities representation. In addition to SNU identity that it is a way of uniquely identifying a member of the crowd:
- Presence (s1) is an MBCE of knowing who is online.
- Relationships (s2) are an MBCE of describing how two members in the development phase are related such as Flickr people who can be contacted and friends of the family.
- Conversations (s3) are an MBCE of messaging to other members through the system.
- Groups (s4) are an MBCE of forming communities of specific work.

- Reputation (s5) is an MBCE of knowing the status of another member of the system such as who's a good member? Who can be trusted?
- Sharing (s6) is an MBCE of sharing things that are meaningful to participants like photos or business model. Due to time and space constraints, we will present the other six blocks shortly.
- Experiences block (E) is used for activity awareness and analyzing the socio-technical complexity of integrating user experience activities into open-source projects. There are many researchers who discussed differences with dynamics in the open user experiences status related to in the community, decision making, and ways of sharing design information [60]. For example, the useful (e1) facet remains vital and yet the interface-centered methods and perspectives of human-computer interaction.
- Motivation block (M) or designing for motivation can be used for the empowering experience of crowds free of any authority. For example, gamification approaches [32] often reduces the complexity of a well-designed and balanced game to its simplest components, e.g., badges, levels, points, and leaderboards play an important role in the interpersonal platform facet (m1).
- Management block (G) is a set of logical properties that are related to management. For example, when you manage many similar SQL Server environments, you can configure (g5) facet in one instance of SQL Server, copy the state of the facet to a file, and then import that file into another instance of SQL Server as a policy.
- SDLC block (C) has a complex activity that requires a diverse set of stakeholders to communicate and coordinate in order to achieve a successful outcome. For example, a platform titled FeatureIT [61] has the goal of supporting the collaboration between stakeholders throughout the entire SDLC (C1:6). FeatureIT was the result of unifying the theoretical foundations of the multidisciplinary field of computer-supported cooperative work (CSCW) with the paradigm and associated technologies of Web 2.0.
- Task block (T) is based on a broad-based empirical study of 1,000 workers on CrowdFlower [62] is based on representing the six facets related to certain aspects of crowd behavior; the task affinity of workers is effort exerted by workers to solve tasks that require human intervention. These facets play a vital role in efficient task design. For example, interpretation and analysis tasks, e.g., t4 and t5, rely on the wisdom of the crowd to use their interpretation skills during task completion (e.g., choosing the most suitable category for each business rules or categorizing reviews as either positive or negative).

Finally, we can say CCD method and C block should be centered around the meaningful T that people are done in, which requires an explicit representation of worker and task profiles in the S, M, E, L, and G models. Hence, the top management model for managing crowdsourcing software building blocks is that:

$$X = x1^U\ x2^U\ x3^U\ x4^U\ x5^U\ x6^U\ x7\ |time \\ where\ X = \{C, T, S, M, E, L, G\} \tag{8.1}$$

8.4.1.1 A Design Solution

Based on CCD and a human-centric activity [63], usability and HCI approaches have tried to augment human task on two separate planes: first deals with machine interpretation of task and the latter explore interactions. However, CCD method has recently utilized human cognition and memory to generate diverse HBCE streams on specific building blocks, which are mostly easy for humans to solve but remain challenging for MBCE, e.g., algorithms. In this section, we present a descriptive model toward the automatically generating personalized hybrid-CE by implementing a simple solution (see Fig. 8.9), focusing on semantic decomposition [62] which is based on MBCE and HBCE specifications, and then mapping these CEs to usability and HCI model.

8.4.1.2 Semantic Decomposition Model: Main Class, Criteria, and Constructors

One key element of the hybrid-CE class, as Fig. 8.10 shows, is a specific specification for a specific MBCE and HBCE to be designed, developed, or modified. These are the specifications for what MBCE or HBCE is going to be used:

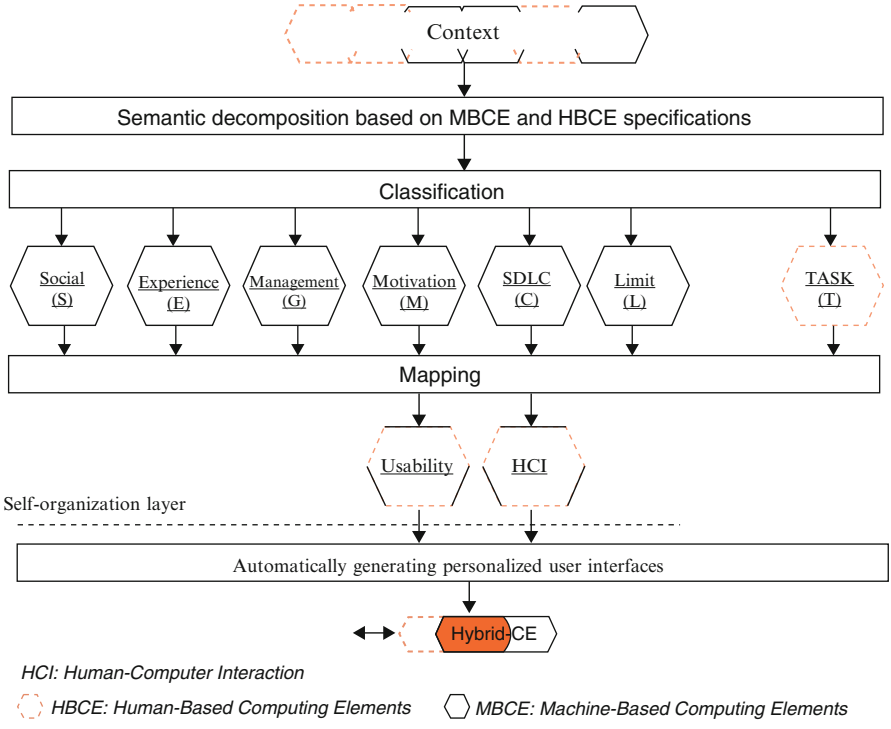


Fig. 8.9 Potential design solution for the hybrid-CE

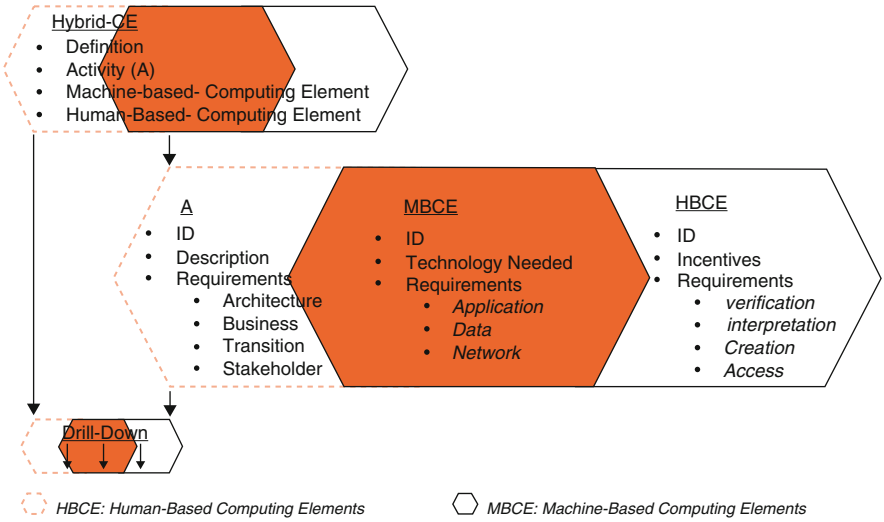


Fig. 8.10 Hybrid-CE class

- From hybrid-CE, we can represent all the MBCE and HBCE embedded in the various activities identified in the development phase.
- From the MBCE specifications, all of the specifications can be appropriately distributed into six aspect categories of IT technologies that link to the three classic IT architectures (data, applications, and technical architecture).
- From the HBCE specifications, all of the specification can be appropriately distributed into one aspect that has six facets (verification, validation, interpretation, analysis, content creation, and content access) [62], which play a vital role in understanding the behavior of workers and their skill and then the HBCE design. For example, a process may require very different actions depending on whether a participant is new or existing with a task achievement history.
- From hybrid-CEs, we can build software specifications, which have all the HBCE required in the various processes identified in the task.

Table 8.1 describes different types of decision unit between CEs that we consider as important criteria, each of which justifies MBCE or HBCE. We consider these criteria as first-class designing elements since CEs are constrained by various types of simplicity, cost, benefit, and quality models. For example, emergency or disaster conditions, e.g., hazards, search and rescue (SAR) is the search for and provision of aid to people who are in distress or imminent danger. In this condition, the signals, which come from the crowd, are very useful. Therefore, an HBCE might be designed to minimize the accuracy of analytics work in order to meet the response time to quickly react to that condition. In addition, it could be designed to utilize inexpensive HBCE to minimize the cost and maximize the accuracy but accept an increasing response time as a trade-off. Therefore, treating

Table 8.1 Different criteria for the decision process for choosing CE type that can be justified in the decision process in terms of simplicity, cost, benefit, and quality

Criterion type	Description
Simplicity	This traditional type of decision indicates how simple a CE is to another. In principle, simplicity can be measured in terms of structure, unstructured, facts, rules parameters, and domain concepts
Decomposition	This well-known criterion of a CE indicates that a CE can be decomposed to MBCE and HBCE
Guidance	A CE guides another CE if the last requires the previous for providing a certain parameter for one of its works
Flow of control	A CE depends on another CE if the outcome of the last decides whether previous should be executed or not
Flexibility	This criterion is used to describe how a CE can be decomposed to the similar task but different cost, benefit, and quality at runtime

these characteristics as first-class designing elements will allow the initiator to explicitly specify, control, and enforce fixable constraints.

In order to assist the development of complex work, we developed a number of high-level design CEs and the relationships between them that help establish interactions among CEs which led to hybrid-CE. Some of the benefits of using the high-level design rather than the low-level design are that it is easier to use and requires less setup. Moreover, the high-level design is self-initializing. Table 8.2 which shows constructs for hybrid-CE have a set of MBCEs that can be called upon the needs; constructs for a relationship have a set of (usage) patterns that can be used to establish the relationship. Constructs for cost, quality, and benefits and constructs for high-level programming interface can focus on the logic of the hybrid-CE, instead of dealing with implementation-specific details of hybrid-CE and complex algorithms for establishing relationships among CEs.

8.4.1.3 Task Collaborator Model

The role of the collaboration model is to translate the hybrid-CE structure as described in previous sections, which drives its actions. In a collaboration model, the MBCE's plan should always be governed by the HBCE, regardless of which CE is performing an activity at any given phase. At its core, the collaboration model is implemented as a state machine (see Fig. 8.11) commanding the SDLC throughout the collaboration and triggering the appropriate social behaviors.

As described in the previous steps, once an activity (Y) is requested as a crowdsourcing, the MBCE is engaged to perform the hierarchical Ys jointly with an HBCE. When an MBCE executes a hierarchical Y, the role of another MBCE is to push the Ys onto a stack and perform each of the actions based on the

Table 8.2 High-level design for relationships in HCE

Construct	Description
Simplicity (HBCE, MBCE, criteria)	True if HBCE is simple to design than MBCE w.r.t. criteria
Flow of data (CE, D, M, HBCE or MBCE)	A CE producing data D which is needed by HBCE or MBCE. The message M_l is the location associated with a CE (e.g., center DB) where the data will be stored and shared
Flow of control (CE_1, CE2, criteria, conx)	A CE_1 depends on another CE_2 if the outcome of last decides w.r.t. criteria whether previous should be executed or not in a given context conx
Guidance (CE_1, $CE2_1$, conx)	Declares that CE_1 should be guided by a CE_2 in a given context conx
Decomposition (CE, HBCE, MBCE)	Creates a use case of HBCE and MBCE for a given CE
Flexibility (CE, [Func], NFPs, CE')	A CE' is a new form of CE to satisfy given simplicity/cost/quality/benefit models

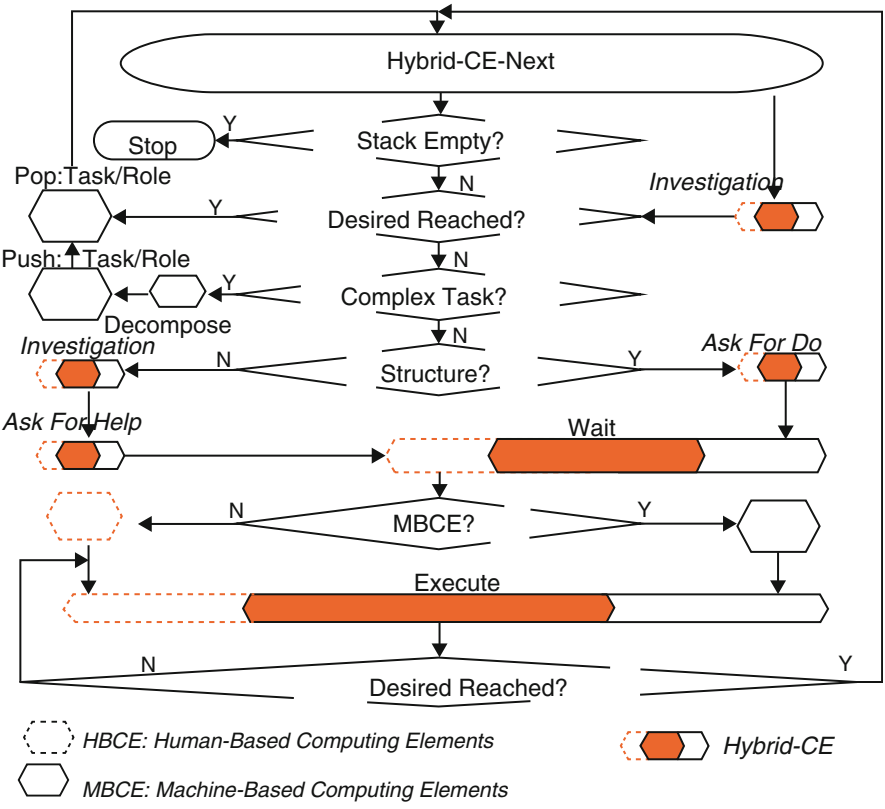


Fig. 8.11 A schematic view of the task collaborator model. Note that the "wait" state can be terminated by both explicit and implicit turn taking on the HBCE

preconditions of the HBCE. The model's states are defined, in flexible order of a typical collaboration, as follows:

- Hybrid-CE-Next: The initial state of the system in which the MBCE evaluates the current state and acts upon this evaluation.
- Ask for do: If an MBCE is capable of achieving the next Y of the development phase, it will offer to take its turn, else next Y the next activity will be executed.
- Ask for help: An MBCE will ask for help from the HBCE, or it will decompose the Y into its constituents, an MBCE and an HBCE, and recursively pushes them to the stack.
- Wait: Waiting for a support from the other CE.
- Execute: If the MBCE is executing the current Y, this happens in this state; if the HBCE is executing a step, the MBCE waits in this state.
- Investigation: Establishing common ground by looking through facts or evidence and come up with a decision.

Moreover, the role of this collaboration model gives us a general view about the interactions between MBCEs and HBCEs. MBCEs interact with HBCEs receiving state information and reinforcement feedback and executing actions. The state information may depend on each other. That is, the next state may depend on current state and on the executed action. Moreover, this interaction is thus more difficult but also more natural, simulating the interaction with an actual outside world. We present the low level of basic collaboration, which we believe will provide additional support to design hybrid-CE.

8.4.2 A Self- Property and a *-Family Property*

A resource for SIC is anything that can be selected by the MBCE. Each resource is represented as $MBCE_{S,E,M,L,G,C}$. For example, let H be a space (honeycomb) that represents all the possible $MBCE_{S,E,M,L,G,C}$ which can be selected for the adaptation hybrid-CE in a moment (e.g., all the possibilities of web pages that can be generated). Each H can be modeled as the union of different aspects which can be adapted (e.g., a web page can be the union of its content, presentation, possible user interactions, and links to web pages). These adaptation blocks can have other different facets (e.g., the links can be external or internal, or the presentation can be divided into color, font, etc.) and so on in a meshing relationship, forming H. This mesh is a graphical representation of all the adaptation blocks which influence the development process. There might be some facets of the H that cannot be further divided into other facets. These indivisible final facets support specific HBCE. For example, let t_i be one of these HBCEs; then t_i can be of the following different types of HBCE:

1. Numerical: It might be an integer, float, etc. A specific example can be a range of specific values for a specific item (e.g., tax slices are 10%, 15%, etc.).

2. String: It might be some adaptive text to show to each worker as social feedback.
3. Categorical: It might be a set of business classes which can be selected.

Therefore, each H is represented as the combination of all its facets for supporting t_i. The different hybrid-CEs for the same H must be disjunctive, so adaptation rules must not include anything of another one (dependency rules). But there can be one type of dependency among one hybrid-CE and other hybrid-CEs (e.g., it is necessary to know the required skill to show to the workers before generating hybrid-CE). This dependency will also be traversed horizontally or vertically to the specific hybrid-CE of each H until the leaf nodes denoted by hybrid-CE_i. Therefore, the hybrid-CE_i can depend on others for supporting its generation. Hence, hybrid-CE_i can be generated through a workflow. There is a set of associated rules, e.g., for determining the next hybrid-CE (e.g., web page content to show to the worker); there can be a rule for filtering hybrid-CE that was already visited by this worker. This filtration is based on if the worker is known or unknown. For example, let be R_i a rule of our engine related to a hybrid-CE_i. A rule can be defined such that as a processing block that receives (e.g., from a file) the information with the possible and available options (H) to select for that support $HBCE_{t:1-6}$ (in general a subset of all the possibilities for t_i). Moreover, hybrid-CE_{i+1} can be given in different formats which also depend on the facet of a block, for example, as candidate options, expressing that are different to a set of MBCEs, in a specific time. The rules of the model can be combined to form new composed rules with the following types of combination rules (Fig. 8.12).

The AE generates rules that are used to generate hybrid-CE, regardless of which CE is performing an activity at any given phase. Then, the hybrid-CE is used as a state machine [5] commanding the SDLC throughout the collaboration and

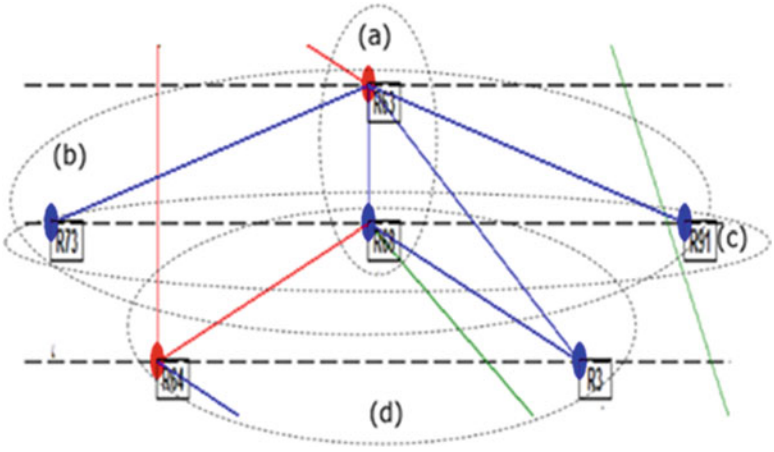


Fig. 8.12 Sequence (**a**), parallel (**b**), union (**c**), and conditional or decision (**d**) rule combinations

triggering the appropriate social behaviors. Hence, the hybrid-CE as an automatically generating personalized user interfaces is based on the following model:

$$
\begin{aligned}
&\textit{If Hybrid-CE} = \textit{HBCE}^{U}\ \textit{MBCE} \mid \textit{time}, \textit{where}: \\
&\textit{HBCE} = T, \textit{and MBCE} = S^{U}\ E^{U}\ M^{U}\ L^{U}\ G^{U}\ C \\
&\textit{And},\ \textit{AE} = \textit{For}\ i = 1\ \textit{to}\ 6 \\
&\qquad\qquad \textit{For}\ j = 1\ \textit{to}\ 6 \\
&\qquad\qquad\qquad \textit{For}\ k = 1\ \textit{to}\ 6 \\
&U\left(\textit{HBCE}_{i,j,k} \cap \textit{MBCE}_{i,j,k} \mid \textit{Constraints} \mid \textit{time-1}\right) \\
&\textit{And}, U\ \textit{is preferences and utility functions} \\
&\textit{And Constraints such as}: \\
&\qquad \textit{Space-of-hybrid-CE} = H^{U}\ \textit{MBCE}_{t:1-6} \\
&\qquad \textit{Dependency-hybrid-CE}_{i} = f\ (\textit{Space-of-hybrid-CE} \mid \textit{Time-1}) \\
&\qquad \textit{Workflow-hybrid-CE}_{i+1} = \textit{hybrid-CEi}\ (\textit{MBCE}_{i,ti}) \\
&\textit{Then}, \\
&\textit{Hybrid-CE} = \textit{AE-Generate}\ \left(\textit{HBCE} \cap \textit{MBCE} \mid \textit{Max}(U) \mid \right. \\
&\left. \textit{Min}(\textit{Constraints}) \mid \textit{time}\right)
\end{aligned}
\tag{8.2}
$$

8.4.3 Hybrid-CE: A Technical Meta-model

In this section, we engineer a hybrid-CE model in more detail, and we position hybrid-CE within a broader taxonomy of detailed meta-model. Taking into account the characteristics of different CEs and the design principles presented above, in this section and the following one, we describe a technical meta-model for its design based on [4]. This meta-model allows specifying the elements of a hybrid-CE and communication flows among them, in a standardized way that allows to easily create various kinds of hybrid-CEs and to modify them when required as shown in Fig. 8.13. This figure follows the traditional application-level description of software systems, with focus on crowdsourcing software. Each CE can be computer driven or human driven (resp. MBCE or MBCE). It performs dedicated tasks, and it may depend on other CEs in terms of data requirements. The CEs are connected through data flow connections. The data flow connections tie input ports and output ports. Data items are emitted from their source (technological node) through a (usually single) output port, and they are used by the target through (possibly multiple) input ports. Communication between CEs is done through generic ports (e.g., input, configuration, and output ports channels). The input ports are used to realize data from different building blocks (aspect and facets as in the next chapter). The configuration ports provide a way to configure a CE or to set default values or derive values for some of its parameters. The output port is used to send the output data of the CE. Each port can have multiple parameters, and the data types of those parameters can be set using either built-in types (primitive data types) or domain-specific types.

A single output port is all that is needed for most applications to send output data. However, in addition to the one-to-one communication between two CEs, it is

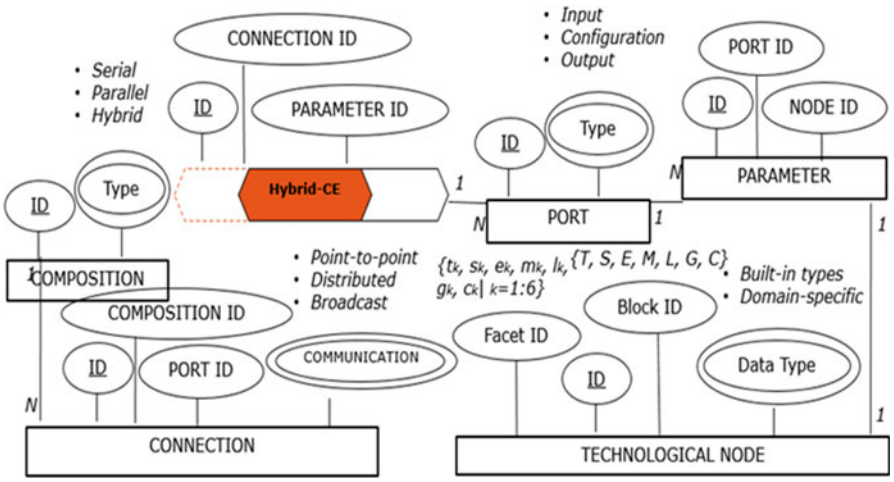


Fig. 8.13 The detailed meta-model of a hybrid-CE, depicting computing elements, connections, and communication flow between CEs and composition types

possible that output data has to be distributed from one source to many target CEs in parallel. To this end, the complexity of determining the destination CEs for each output item can be managed by one of the following communication models:

1. Point-to-point communications are the simplest case and have a single CE such as MBCE or HBCE. Essentially, both source and target CEs in a point-to-point communication model comprise one port each (i.e., source with the output port and target with the input port).
2. The distribution communication model supports cases where the output data of single CE (e.g., data source CE) has to be realized by multiple CEs. In such cases, multiple CEs are subscribed to the same post using the distributed communication model, possibly filtering data according to certain criteria or keys.
3. Broadcasted communication model is used when multiple CEs need to process the same data items at the same time. It duplicates data items to all consumers that are subscribed to a port.

A composition model represents a specific topology used to connect various CEs in order to solve a specific problem. Hybrid-CE can be composed using mainly the three composition modeled such as:

1. A serial composition that represents two or more CEs connected in a serial way, according to which the target CE always depends on the source CE in terms of data or parameters.
2. A parallel composition that represents the case where the connected CEs work in parallel and are independent of each other in terms of data or parameters.
3. The hybrid composition represents a mix of serial and parallel types and often a more complex composition of CEs.

8.5 Hybrid-CE Design: Applications Example

To evaluate our CrowdSWD, we designed 34 hybrid-CEs or heterogeneous computing elements that can be used in crowdsourcing software. For simplicity purposes, we followed the usability approaches for naming them as given below:

$HC = Honeycomb\ model$, $H_i = Hybrid\text{-}CE_i$, $Des = Description$, $MB = MBCE$, $HB = HBCE$, *and* $BB = Building\text{-}blocks$

- HC: Paper prototyping (H1).
  - Des: It is an approach that supports the entire development process with crowd-centered activities, in order to create applications which are easy to use and added value to the intended SNUs, e.g., getting started.
  - MB: Show some sort of standards for the way an interface is going to look and feel. Present a consistent picture with a guideline to the SNUs and doesn't commit some of the most elementary mistakes. Invite other SNUs into the discussions.
  - HB: Raise awareness about business topics in the rest of the development SNUs by engaging them.
  - BB: Planning, $T^{U}C^{U}S^{U}M$.
- HC: Discussion board (H2).
  - Des: Collect information about the purpose of the system and its overall context of use, e.g., SNUs meeting.
  - MB: Bring together all available SNUs relevant to the development in order to create a common vision. Collect and agree on detailed information about the context of use.
  - HB: Identify the key issues that are needed to explore. Provide all participants with a copy of a list of the issues to be discussed at the meeting such as why is the system being developed? What are the overall objectives? How will it be judged as a success? What are the technical and environmental constraints? Are there any initial design concepts? Is there an existing or competitor system?
  - BB: $T^{U}C^{U}E$.
- HC: A detailed checklist (H3).
  - Des: Collect and agree detailed information about who are the intended SNUs, e.g., analyzes context of use such as what are their tasks? Why will they use the system? What are their experience and expertise?
  - MB: Collect the information to arrange about SNUs who have knowledge about the intended context and then invite them.
  - HB: Fill in each item on the context checklist.
  - BB: $T^{U}C^{U}E$.

- HC: To do the task (H4).
  - Des: Means of managing individual usability activities as well as the overall role played by usability input within a software engineering program.
  - MB: A task manager is appointed for each task, an appropriate activity is selected, and a schedule is specified. Enables priorities to be assessed and facilitates the efficient allocation of resources.
  - HB: For each selected aspect or facet of the system, discuss what sort of targets could be set and the work that needs to be done to achieve those targets.
  - BB: $T^{U}C^{U}G$.
- HC: Use case (H5).
  - Des: A general set of actions you will do with each case on the context of use analysis. Standardized user satisfaction questionnaire is recommended. It identifies the strengths and weaknesses of competing products or services before starting work on prototypes, e.g., competitor analysis [64].
  - MB: Search the web using at least ten different search engines with this set of keywords. Make a list of the top ten sites from the results of the different search engines.
  - HB: Set of terms is most probably too vague. Ask domain experts.
  - BB: $T^{U}C^{U}M^{U}G^{U}E$.
- HC: A list of choices (H6).
  - Des: Generating a list of choices, open-ended questions, or a "walk-through" of the survey with a small number of typical respondents based on aspects of the context of use and what aspects pose problems or raise uncertainties, e.g., by placing hit counters on a web site and by setting the patterns of work. It is a means of finding out how the task is likely to be used by a specific set of SNUs and who these users are likely to be, e.g., user survey for design.
  - MB: Sampling theory is complex, for example, counting of frequencies of response to options or coding and tabulating the data should be as automatic as possible and then sending a warning. These are done by presenting results through giving the headline and then the detailed analysis of how SNUs got there and finish with a conclusion based on the data.
  - HB: Interpret or ask others what they understand by each question as they go through the survey or by forming own opinions or biases as you work with the data.
  - BB: Requirement through $T^{U}C^{U}E^{U}$ $S^{U}L$.
- HC: Indicative table of contents (H7).
  - Des: Discovering facts and opinions held by potential SNUs of the system being designed, e.g., interviews.
  - MB: In order of preference, record the informant's responses.

HB: Address directly the informant's individual concerns, e.g., askable prompt. Check with the intended SNUs that this is useful for them.
BB: $T^{U}C^{U}M^{U}S^{U}L$.

- HC: Task diagrams (H8).
  Des: Gathering field data from users, e.g., contextual inquiry [65].
  MB: Watching users do their work and interacting with colleagues, which doesn't steal much time from the SNUs. Recording and organizing tasks. Automatic renumbering of tasks and plans after each edit.
  HB: Connect with the person ordering the contextual inquiry. This is done by watching and occasionally interrupting, and then building the story together with him.
  BB: $T^{U}C^{U}G$.

- HC: Pilot observation (H9).
  Des: It means a session that involves informal activities with the general public.
  MB: Viewing SNUs as they work in a field study and taking notes on the activity that takes place, e.g., user observation and field studies [66].
  HB: Gain cooperation of contacts with the observation technique that you intend to carry out. Establish the times, places, and SNUs who will be observed.
  BB: $T^{U}C^{U}G^{U}M^{U}S$.

- HC: A template format (H10).
  Des: A detailed checklist that provides a basis for designing. Collecting similar information about SNUs and what are their tasks? What are the technical and environmental constraints? For example, analyze the context of use.
  MB: Collecting similar information about the users and what are their tasks? What are the technical and environmental constraints? Invite members who have experience in the business.
  HB: A description of the context of use derived from the completed checklist.
  BB: $T^{U}C^{U}E^{U}S$.

- HC: Focus group sessions, card sort, affinity diagramming, and prototyping activities (H11).
  Des: An informal assembly of users whose opinions are requested about a specific topic, e.g., focus groups [67]. A focus group is an informal assembly of users whose opinions are requested about a specific topic. The goal is to elicit perceptions, feelings, attitudes, and ideas of participants about the topic. Focus groups are not generally appropriate for evaluation. SNUs come together and express diverse views on the topic: useful not only to find the range of views but also for the

participants to learn from each other and to generate a sense of social cohesion.

MB: Preparing a script or list of issues which need to be tackled. Selecting all the participants from the same category or neighborhood. Inviting to each focus group session with an explanation. Generating hypotheses.

HB: Express the unique points of view. Participate in the context of use analysis or brainstorming.

BB: $T^{U}C^{U}E^{U}S^{U}L$.

- HC: A whiteboard (H12).
  Des: Generate group creativity, e.g., brainstorming. A group of SNUs come together and focus on a problem or proposal.
  MB: Recount and arrange the clusters of ideas that have been formed and then invite the participants to take turns to present one and only one idea at a time.
  HB: SNUs write down their ideas on their own Post-its at any time.
  BB: $T^{U}C^{U}E^{U}M$.
- HC: A list problems (H13).
  Des: Obtain baseline measures of development, e.g., evaluate existing system.
  MB: Collecting most important tasks and SNUs to be tested (based on the context of use study). Generate a paper prototype for testing.
  HB: Categorized by importance, e.g., use Post-it notes to sort the problems.
  BB: $T^{U}C^{U}E$.
- HC: Tasks board (H14).
  Des: Discovering the latent structure in an unsorted list of statements or ideas, e.g., card sorting/affinity diagramming [68]. Sort large amounts of data into a logical group.
  MB: Collecting a large amount of data based on the context. Identify and group SNUs' functions as part of design.
  HB: Sort these data into categories.
  BB: $T^{U}C^{U}E^{U}S$.
- HC: An activities outline (H15).
  Des: How SNUs carry out their tasks in a specified context, e.g., scenarios of use (use cases) [69], such as users' activities, goals, and motivations for using the system being designed, and the tasks they will perform.
  MB: Identify intended SNUs, their tasks, experiences, and the general context.
  HB: Specify how users carry out their tasks in a specified context.
  BB: $T^{U}C^{U}E^{U}M^{U}L$.
- HC: Layered diagram (H16).
  Des: It means a check for consistency. Analyzes what a SNU is required to do in terms of actions and/or cognitive processes to achieve a task, e.g., task analysis [70].
  MB: Draw the subtasks as a layered diagram ensuring that it is complete.

HB: Break tasks down into subtasks in terms of objectives, e.g., task decomposition.
BB: $T^{U}C^{U}E^{U}G$.

- HC: Tasks objectives (H17).
  Des: A workshop attended by SNUs who identify system requirements that can be tested later in the development process, e.g., requirements meeting [71].
  MB: Indent SNU and tasks in the context of use analysis.
  HB: Provides functional and nonfunctional criteria that can be tested.
  BB: $T^{U}C^{U}E^{U}S$.

- HC: Drag and drop design items (H18).
  Des: Design guidelines [72] for user task design, summarize good practice, and provide useful high- and low-level guidance on the design of task.
  MB: Collect guidelines for creating well-designed functionally.
  HB: Drag and drop design items.
  BB: Design – $T^{U}C^{U}E^{U}L$.

- HC: Activity on a Post-it note (H19).
  Des: Paper prototypes [73] or other mock-ups are used to clarify requirements to be very rapidly simulated and tested.
  MB: Collect the domain concepts.
  HB: Design and sketch out possible concepts in a brainstorming environment. Carry out a realistic task, e.g., based on the context of use scenarios.
  BB: $T^{U}C^{U}E$.

- HC: A list of identified problems (H20).
  Des: It is a heuristic evaluation [74] in a form of development inspection where development specialists judge.
  MB: Generate mock tasks and record SNU's observations.
  HB: Feedback (recommendations for design improvements) on the problematic aspects to designers.
  BB: $T^{U}C^{U}E^{U}L$.

- HC: Basic prototypes (H21).
  Des: Parallel design [75] is a method where alternative designs, often task designs, are created by SNUs at the same time independently.
  MB: Collect SNUs to be available concurrently in order to carry out design work in parallel. Collect a requirements document, criteria, and boundaries which are needed to make sure that the design SNUs are given the same information so that design work starts from the same starting point.
  HB: Discuss with other different aspects of the designs.
  BB: $T^{U}C^{U}E^{U}L$.

- HC: Lists and charts with time and resource constraints (H22).
  Des: A storyboard [76] is a low-fidelity prototype consisting of a series of tasks and sketches. Provides an overview of the system. Demonstrates the functionality of the storyboard elements. Demonstrates the navigation scheme.
  MB: Generate the context of use and scenarios.
  HB: Select the best storyboard elements.

- HC: Storyboards prototypes (H23).
  Des: Prototype to identify development problems, where the SNU is probed to explain their expectations and problems, e.g., evaluate the prototype.
  MB: Generate the most important tasks and SNUs to be tested. Plan sessions allowing time for giving instructions and running the test.
  HB: Select the best storyboard elements.
  BB: $T^{U}C^{U}E^{U}M$.

- HC: Simulation (H24).
  Des: System's responses in real time. The Wizard technique enables unimplemented technology to be evaluated by using a human to simulate the response of a system.
  MB: Gather information about the nature of the interaction.
  HB: User's input.
  BB: $T^{U}C^{U}L$.

- HC: Design patterns (H25).
  Des: The SNUs conceptualize the task in terms of interface design patterns or pattern-based design [77] during requirements elicitation itself.
  MB: Generate design pattern examples.
  HB: Express business concepts in terms of design patterns.
  BB: $T^{U}C^{U}E^{U}M$.

- HC: Visualizing (H26).
  Des: Visualize a part of business mode. Style guides are used to provide the consistency between tasks.
  MB: Generating a well-designed, visually, and functionally consistent business model.
  HB: Comments in terms of understandability.
  BB: Implementation through $T^{U}C^{U}E$.

- HC: HyperCard and ToolBook (H27).
  Des: Visualize a part of the system. Rapid prototyping or interactive prototypes [78] are developed which can be quickly replaced or changed in line with design feedback.
  MB: Generate different proposed concepts, and prepare realistic tasks.
  HB: Provide feedback on it.
  BB: $T^{U}C^{U}E^{U}M^{U}L$.

- HC: Task scenarios (H28).
  - Des: User-based evaluation of a working system, where the primary objective is to identify development problems, e.g., diagnostic evaluation [79]. Input data and write instructions for the user (tell the user what to achieve, not how to do it).
  - MB: Collect the most important users, tasks, and environment that are used. Generate instructions for the SNUs (tell the user what to achieve, not how to do it).
  - HB: Interpretation of the contents of each task scenarios and their reason for making choices.
  - BB: Testing through $T^{U}C^{U}E$.
- HC: Performance testing (H29).
  - Des: Performance testing [80] is a rigorous evaluation of a working system under realistic conditions to compare measures such as success rate, task time, and user satisfaction with requirements.
  - MB: Ensure that the SNUs, tasks, and environment used for the test are representative of the intended context of use. Invite developers to observe the task scenario.
  - HB: Explain the interpretation of the contents of each task scenario and the reason for making choices.
  - BB: $T^{U}C^{U}S^{U}L$.
- HC: General-purpose screening (H30).
  - Des: The usual method of assessment is to use a standardized opinion questionnaire to avoid criticisms of subjectivity, e.g., subjective assessment (testing and post-release) [81].
  - MB: Collect some background data about the context of use, e.g., computer experience, job level, and the frequency of use of the software being evaluated.
  - HB: Fill out a questionnaire.
  - BB: $T^{U}C^{U}L$.
- HC: Survey from the start (H31).
  - Des: SNUs are asked to identify specific incidents which they experienced personally and which had an important effect on the final outcome, e.g., critical incident technique analysis [82].
  - MB: Collect a sufficient quantity of data in which you should be able to categorize the incidents and produce a relative importance weighting for each. Content analysis in order to summarize the experiences of many users or many experiences of the same user.
  - HB: Describing the positive and negative critical incidents.
  - BB: $T^{U}C^{U}L$.

- HC: Questionnaires (H32).
  Des: Evaluate the pleasure of the product [83]. The general idea then is that a more holistic approach on the relationship between the user and the machine will give added value to SNU's pleasure with the product.
  MB: Collect data about time on task, the number of errors, and subjective opinion.
  HB: Evaluate usability of the product.
  BB: $T^{U}C^{U}M^{U}S$.
- HC: Casebook (H33).
  Des: It must be understood that all measurement involves sampling, e.g., post-release testing and measurement.
  MB: Collect large samples until the product is released.
  HB: Vote on which may become an incentive to increase general SDLC.
  BB: Post-release through $T^{U}C^{U}E^{U}M^{U}S$.
- HC: Spreadsheet (H34).
  Des: Is a very useful tool for keeping raw survey results with some kind of reward. User surveys for design are a means of finding out how the task is likely to be used by a specific set of users, and who these users are likely to be.
  MB: Collect a large number of SNUs; usage profiles from user surveys can be relied upon and analyzed statistically, and this gives moderately hard objective data.
  HB: Post must be relevant to the issues that are important to the design team such as the major decision points or areas of uncertainty.
  BB: $T^{U}C^{U}E$.

8.6 Experimental Study

To validate CrowdSWD, we initiated an experimental study of a software development application to build a payroll system. We chose to develop a business model helping the crowd to generate multiple business rules. The system will allow a member of the crowd (crowdsourcers) to sign up for an account, post an abstract model, and ask the other member (providers) to decompose this model into slices (shape and concepts). The abstract model was a number of images and symbols. This is followed by a precise technical illustration of business issues, which are required for management. This was the first phase (posting the crowdsourcer requirements) and so on to the next phases of SDLC. In each development phase, CrowdSWD used five hybrid-CEs (artifacts) to address design problems (see Fig. 8.14). The crowds were employees who have different skills, experiences, and other facets of the previous building blocks. After registration process (case 1) or login process (case 2), adaptation engine (AE) creates worker profile (in case 1).

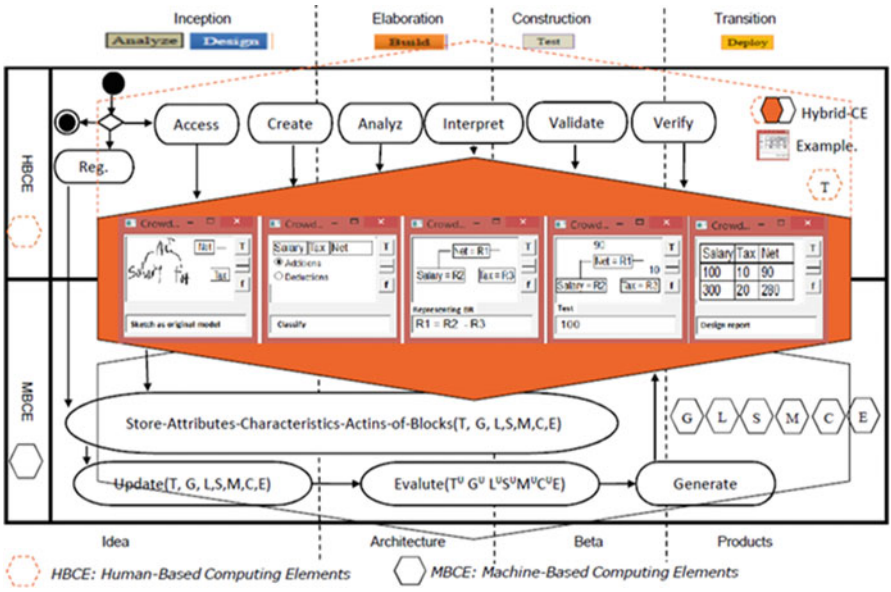


Fig. 8.14 Hybrid-CE model for crowdsourcing software (payroll software)

Then, this profile is matched with the seven building blocks by AE (in case 1, 2). Based on the result, AE generates hyper-CE as an interface for the worker. This hybrid-CE contains the task and some facilities (MBCE) for achieving those tasks. It also contains a simple request or human intelligence tasks (HITs) such as "Please sketch and type the concept as they are in the original sketch!". Each action is taken, stored, and shown to other workers. At the same time, AE updates the knowledge base and worker profile. Based on worker profile, AE collects and agrees on detailed information about the intended workers and what are their HBCE?

8.7 Experimental Results and Discussion

Based on the results of the experiment, all types of hybrid-CEs were executed; the work was very simple (e.g., the child can share). Over time, the participation rate increases and goes up. The success criteria are evaluated (fulfilling the triple constraints mean that the project is perceived as successful) and checked through an expert. The experiment generated interesting results. Of the 140 CE posted, only 120 were successfully completed. The overall complexity of developing the system through our model is well comparable with that of traditional development [5]. The results of the experiment are encouraging, and there are multiple tracks for the further crowd work. Now the great part is that after some time, AE can see which one is more successful. AE learned something. AE never stops doing this through your whole SDLC, forever. The customized findings and recommendations based on the

Table 8.3 Recommendations for your general SDLC

AE	Building blocks		Hybrid-CE	Development phase
	MBCE	HBCE		
Generating and update	Task allocation	Technical infeasibility	–	AE decision
	Alternative designs	Create	Parallel design [76]	Design DesignL
	Alternative options	Selects options	Evaluate	Test, design, req., $T^UC^UM^US$
	Relevant contextual information	Awareness	Expert evaluation	Post-release $T^UC^US^UL$
	Problems and situation	Awareness	Patterns	Design $T^UC^UE^UL$
	Present data	Sort them into logical groups	Affinity diagramming	Req. $T^UC^UM^US$
	Whiteboard	See what ideas have been generated	Brainstorming	Req. $T^UC^UL^US$
	Unsort list and stick the labels on index cards	Discovering	Card sorting	Req. $T^UC^UM^US$
	Gathering field data	Ask others	Contextual inquiry	Req. analysis $T^UC^UM^US^UL$
	Visualizing ontology	Concept design	Prototype	Design $T^UC^US^UM$
	Success rate	Realistic conditions	Testing	Test T^UC^UL
	Test the system concept	No human factors expertise	Scenarios	Design $T^UC^US^UM$
	Simulation	Interaction	Storyboarding	Design, impl. $T^UC^UL^UM$
	Observes the user's actions and simulates the system's responses	Unaware and training	Wizard	Design, imp. $T^UC^US^UM$

experiment, organizational constraints, and goals were 14 hybrid-CEs (see Table 8.3) which might be suitable recommendations for generalized crowdsourcing SDLC.

8.8 Conclusion

This chapter explored one instance of SDLC called CrowdSWD. It included a range of activities, entities, processes, and contexts including (1) initiation of the standards development effort, (2) the stages of the development of this standard, and (3) the interested parties and their interaction. The goals and objectives of this chapter developed a holistic understanding of CrowdSWD through (1) design and

manage the context within CrowdSWD occurred and discover the important aspects that enable or constrained its design and management, (2) unify the preliminary conceptual model of hybrid-CE to reflect CrowdSWD, and (3) explore other information technology standards development efforts. The methodology was reasoning and learning about characteristics of participants. The output was mechanisms for task routing and problem-solving that harness participant's expertise. Basic components were (1) modeling crowdsourcing software and human tasks that empower a crowd socially to solve complex problems that require effective management among participants with relevant abilities and limitations, (2) modeling supportive environments for crowdsourcing software, and (3) modeling adaptive engine that learns relevant characteristics of participants based on observations of their behavior and learned models. The single experimental study, reported in this chapter, provides richness of data that can lead to mainly answer two questions: (1) What are the aspects, building blocks, entities, processes, and mechanisms that influence, enable, or constrain CrowdSWD? and (2) What are the different hybrid-CEs that reflect CrowdSWD?

References

1. Shi Z, He H, Luo J, Lin F, Zhang H (2006) Agent-based grid computing. Appl Math Model 30 (7):629–640
2. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. In: Grid computing environments workshop, GCE '08. IEEE, Piscataway, pp 1–10
3. Ali T, Nasr ES, Gheith M (2014) Socially intelligent computing—a survey of an emerging field for empowering crowd. In: Proceedings of the 9th international conference on INFOrmatics and Systems (INFOS). IEEE, Piscataway
4. Imran M, Lykourentzou I, Naudet Y, Castillo C (2014) Engineering crowdsourced stream processing systems. arXiv preprint arXiv
5. Ali T, Gheith M, Nasr ES (2016) CrowdCE: a collaboration model for crowdsourcing software with computing elements. Int J Recent Innov Trends Comput Commun (IJRITCC) 4 (2):204–213
6. Ackoff RL (1971) Towards a system of systems concepts. Manag Sci 17(11):661–671
7. Von Bertalanffy L (1972) The history and status of general systems theory. Acad Manag J 15 (4):407–426
8. Geiger D (2015) Personalized task recommendation in crowdsourcing systems. Springer, Cham
9. Geiger D, Rosemann M, Fielt E, Schader M (2012) Crowdsourcing information systems-definition, typology, and design. In: Proceedings of the 33rd international conference on information systems. Orlando
10. Burégio V, Meira S, Rosa N (2013) Social machines: a unified paradigm to describe social web-oriented systems. In: Proceedings of the 22nd international conference on world wide web companion. p 886–889
11. Scekic O, Truong H-L, Dus S (2013) Incentives and rewarding in social computing. Commun ACM 56(6):72–82
12. Salehie M, Tahvildari L (2009) Self-adaptive software: landscape and research challenges. ACM Trans Auton Adapt Syst (TAAS) 4(2):1–42

13. Boehm BW, Brown JR, Lipow M (1976) Quantitative evaluation of software quality. In: The 2nd international conference on software engineering. IEEE Computer Society Press, Washington, DC
14. Ralph P (2015) The sensemaking-coevolution-implementation theory of software design. Sci Comput Program 101:21–41
15. Murray-Rust D, Robertson D (2015) "Bootstrapping the Next Generation of Social Machines." Crowdsourcing: crowdsourcing. Springer, Berlin
16. Estellés-Arolas E, González-Ladrón-de-Guevara F (2012) Towards an integrated crowdsourcing definition. J Inf Sci 38(2):189–200
17. Wang F-Y, Zeng D, Carley KM, Mao W (2007) Social computing: from social informatics to social intelligence. IEEE Intell Syst 22(2):79–83
18. King I, Jiexing L, Kam CT (2009) A brief survey of computational approaches in social computing. In: Proceedings of the international joint conference on neural networks. Atlanta. p 1625–1632
19. Shaw A (2012) Using chatbots to teach socially intelligent computing principles in introductory computer science courses. In: Proceedings of the 9th international conference on information technology. Las Vegas. p 850–851
20. Wang F-Y, Zhang D, Sycara K (2013) Guest editorial: special section on social and economic computing. IEEE Trans Serv Comput 6(2):150–151
21. Horvath L, Rudas IJ, Bit JF, Hancke G (2005) Intelligent computing for the management of changes in industrial engineering modeling processes. In: Proceedings of the IEEE 3rd international conference on computational cybernetics. IEEE, Piscataway, pp 249–254
22. Demirbas M, Bayir MA, Akcora CG, Yilmaz YS (2010) Crowd-sourced sensing and collaboration using twitter. In Proceedings of the IEEE international symposium on a world of wireless mobile and multimedia networks. Montrreal, pp 1–9
23. Vukovic M (2009) Crowdsourcing for enterprises. In: Proceedings of the world conference on services – I. Los Angeles, pp686–692
24. Gomes C, Schneider D, Moraes K, de Souza J (2012) Crowdsourcing for music: survey and taxonomy. In: Proceedings of the IEEE International conference on Systems, Man, and Cybernetics (SMC). Seoul, pp 832–839
25. Gonnokami K, Morishima A, Kitagawa H (2013) Condition-task-store: a declarative abstraction for microtask-based complex crowdsourcing. In: Proceedings of the 1st VLDB workshop on databases and crowdsourcing, pp 20–25
26. Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I (2010) Pregel: a system for large-scale graph processing. In: Proceedings of the ACM SIGMOD international conference on management of data
27. Breslin J, Decker S (2007) The future of social networks on the internet: the need for semantics. Internet Comput 11(6):86–90
28. Datta A, Buchegger S, Vu LH, Strufe T, Rzadca K (2010) Decentralized online social networks. In: Handbook of social network technologies and applications. Springer, New York
29. Han B, Pan H, Anil Kumar VS, Marathe MV (2012) Mobile data offloading through opportunistic communications and social participation. Mob Comput 11(5):821–834
30. Cheng CD, Kosorukoff A (2004) Interactive one-max problem allows to compare the performance of interactive and human-based genetic algorithms. In: Genetic and evolutionary computation conference. Springer, Berlin/Heidelberg
31. Liu F, Lee HJ (2010) Use of social network information to enhance collaborative filtering performance. Expert Syst Appl 37(7):4772–4778
32. Deterding S (2012) Gamification: designing for motivation. Interactions 9(14):14–17
33. Deterding S, Sicart M, Nacke L, O'Hara K, Dixon D (2011) Gamification. Using game-design elements in non-gaming contexts. CHI'11 extended abstracts on human factors in computing systems
34. von Ahn L, Blum M, Hopper NJ, Langford J (2003) CAPTCHA: using hard ai problems for security. Adv Cryptol EUROCRYPT 2656:294–311

35. Louis B. Rosenberg (2015) Human swarms, a real-time paradigm for collective intelligence. Collective Intelligence
36. Sanchez-Charles D, Muntes-Mulero V (2015) CrowdWON: a modelling language for crowd processes based on workflow nets. AAAI
37. Barowy DW, Curtsinger C, Berger ED, McGregor A (2012) Automan: a platform for integrating human-based and digital computation. In: Proceedings of the ACM international conference on object oriented programming systems languages and applications, vol 47. ACM, New York, pp 639–654
38. Dwarakanath A, Chintala U, Shrikanth NC, Virdi G, Kass A, Chandran A, Sengupta S, Paul S (2015) CrowdBuild: a methodology for enterprise software development using crowdsourcing. In: Proceedings of the second international workshop on crowdsourcing in software engineering. IEEE Press, Piscataway, pp 8–14
39. Gajosa KZ, Welda DS, Wobbrock JO (2010) Automatically generating personalized user interfaces with supple. Artif Intell 174(12):910–950
40. Amsterdamer Y, Kukliansky A, Milo T (1968) NL2CM: a natural language interface to crowd mining. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2015
41. Bernstein MS, Little G, Miller RC, Hartmann B, Ackerman MS, Karger DR, Crowell D, Panovich K (2015) Soylent: a word processor with a crowd inside. Commun ACM 58 (8):85–94
42. Wu W, Tsai W-T, Li W (2013) Creative software crowdsourcing: from components and algorithm development to project concept formations. Int J Creat Comput 1(1):57–91
43. https://www.topcoder.com
44. Kulkarni AP, Can M, Hartmann B (2011) Turkomatic: automatic recursive task and workflow design for mechanical turk. Human factors in computing systems, pp 2053–2058
45. Franklin MJ, Kossmann D, Kraska T, Ramesh S, Xin R (2011) Crowddb: answering queries with crowdsourcing. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, New York, pp 61–72
46. Ahmad S, Battle A, Malkani Z, Kamvar S (2011) The jabberwocky programming environment for structured social computing. In: Proceedings of the 24th annual ACM symposium on User interface software and technology. ACM, New York
47. Minder P, Bernstein A (2012) Crowdlang: a programming language for the systematic exploration of human computation systems. In: Social informatics, pp 124–137
48. Dustdar S, Schall D, Skopik F, Juszczyk L, Psaier H (2011) Socially enhanced services computing: modern models and algorithms for distributed systems. Springer Science & Business Media, Wien
49. Dustdar S, Truong H-L (2012) Virtualizing software and humans for elastic processes in multiple clouds–a service management perspective. Int J Next Gener Comput (IJNGC) 3:109–126
50. van Ham F, Schulz H-J, Dimicco JM (2009) Honeycomb: visual analysis of large scale social networks. Human-Computer Interaction–INTERACT, pp 429–442
51. Burégio VA, Meira SL, Rosa NS, Garcia VC (2013) Moving towards "Relationship-Aware" applications and services: a social machine-oriented approach. In: Proceedings of 17th IEEE International on Enterprise Distributed Object Computing Conference Workshops
52. Pereira R, Baranauskas MCC, da Silva SRP (2010) Social software building blocks: revisiting the honeycomb framework. In: International Conference on Information Society (i-Society)
53. Kittur A, Nickerson JV, Bernstein MS, Gerber EM, Shaw A, Zimmerman J, Lease M, Horton JJ (2013) The future of crowd work. In: Proceedings of the conference on computer supported cooperative work, New York. ACM, New York, pp 1301–1318
54. Malone TW, Laubacher R, Dellarocas C (2009) Harnessing crowds: mapping the genome of collective intelligence. Boston University, Boston
55. Falkenberg ED, Hesse W, Lindgreen P, Nilsson BE, Han Oei JL, Rolland C (1998) A framework of information systems. The FRISCO Report (Web edition)

56. Alter S (2010) Bridging the chasm between sociotechnical and technical views of systems. In: Proceedings of the 31st international conference on information systems. St. Louis
57. Rhodes DL (2012) The Systems Development Life Cycle (SDLC) as a standard: beyond the documentation. US Census Bureau, Washington, DC, pp 194–2012
58. Neumeyer L, Robbins B, Nair A, Kesari A (2010) S4: distributed stream computing platform. In: Proceeding of IEEE international conference on data mining workshops, pp 170–177
59. Bozzon A, Fraternali P, Galli L, Karam R (2014) Modeling crowdsourcing scenarios in socially-enabled human computation applications. J Data Semant 3(3):169–188
60. Bach PM, Carroll JM (2010) Characterizing the dynamics of open user experience design: the cases of firefox and OpenOffice. org. J Assoc Inf Syst 11(12):902–925
61. Siller GG (2013) FeatureIT: a platform for collaborative software development
62. Gadiraju U, Kawase R, Dietze S (2014) A taxonomy of microtasks on the web. In: Proceedings of the 25th ACM conference on hypertext and social media. ACM, New York
63. Mao K, Capra L, Harman M, Jia Y (2015) A survey of the use of crowdsourcing in software engineering. RN 15(1)
64. Sierzchula W, Bakker S, Maat K, van Wee B (2012) The competitive environment of electric vehicles: an analysis of prototype and production models. Environ Innov Soc Transit 2:49–65
65. Druin A (1999) Cooperative inquiry: developing new technologies for children with children. In: Proceedings of the SIGCHI conference on human factors in computing systems
66. Malinen S (2015) Understanding user participation in online communities: a systematic literature review of empirical studies. Comput Hum Behav 46:228–238
67. Carolan M, Holman J, Ferrari M (2015) Experiences of diabetes self-management: a focus group study among Australians with type 2 diabetes. J Clin Nurs 7–8(24):1011–1102
68. Widjaja W, Sawamura M (2015) DADS system: distributed approach to digital affinity diagram collaboration. In: Proceedings of the 18th ACM conference companion on computer supported cooperative work & social computing
69. Adams B, Kavanagh R, Hassan AE, German DM (2016) An empirical study of integration activities in distributions of open source software. Empir Softw Eng 21(3):960–1001
70. Prommann M, Zhang T (2015) Applying hierarchical task analysis method to discovery layer evaluation. Inf Technol Libr (Online) 44(1)
71. Mylopoulos J, Chung L, Liao S, Wang H (2001) Exploring alternatives during requirements analysis. IEEE Softw 18(1):92–96
72. Maguire M (2001) Methods to support human-centred design. Int J Hum Comput Stud 55 (4):587–634
73. Beyer H, Holtzblatt K (1999) Contextual design. Interactions 6(1):32–42
74. Manning MD, Harriott CE, Hayes ST, Adams JA, Seiffert AE (2015) Heuristic evaluation of swarm metrics' effectiveness. In: Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction extended abstracts
75. Xu A, Rao H, Dow SP, Bailey BP (2015) A classroom study of using crowd feedback in the iterative design process. In: Proceedings of the 18th ACM conference on computer supported cooperative work & social computing
76. Newman MW, Landay JA (2000) Sitemaps, storyboards, and specifications: a sketch of web site design practice. In: Proceedings of the 3rd conference on designing interactive systems: processes, practices, methods, and techniques
77. Thanh-Diane N, Jean V, Ahmed S (2016) Generative patterns for designing multiple user interfaces. 3rd IEEE/ACM International Conference on Mobile Software Engineering and Systems MobileSoft
78. Fitton D et al (2005) Rapid prototyping and user-centered design of interactive display-based systems. IEEE Pervasive Comput 4(4):58–66
79. Bengtsson C (2015) End-to-end set-up of crowdsourced evaluation of utterance clusters in big data. Uppsala University, Ed.: Master's Thesis in Computational Linguistics
80. Yuen M-C, King I, Leung K-S (2015) Taskrec: a task recommendation framework in crowdsourcing systems. Neural Process Lett 41(2):223–238

81. Raza A Capretz LF (2015) Contributors preference in open source software usability: an empirical study. arXiv preprint arXiv:1507.06882
82. Marsden J (2013) Stigmergic self-organization and the improvisation of Ushahidi. Cogn Syst Res 21:52–64
83. Schneider H, Frison K, Wagner J, Butz A (2016) CrowdUX: a case for using widespread and lightweight tools in the quest for UX. In: Proceedings of the 2016 ACM conference on designing interactive system

Chapter 9
An Approach to Migrate and Manage Software: Cloud-Based Requirements Management

Areeg Samir

9.1 Introduction

Software industry is considering cloud computing to increase its working competency through using cloud's main promised characteristics such as on-demand service, rapid elasticity, anywhere and anytime data access, and resource pooling. Gartner [1] defines cloud computing as "a style of computing where massively scalable IT-enabled capabilities are delivered 'as a service' to external customers using Internet technologies." There are many approaches to cloud services deployment, viz., Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Cloud computing promises to reduce IT services costs, better management efforts, and clear maintenance responsibilities to allow enterprises to focus more on production and innovation. It increases business agility and scalability through permitting enterprises to satisfactorily meet the needs of the rapidly changing environments. Cloud significant features support an increasing number of enterprises which view it as a key of business transformation that attract customer engagement, establish new partnerships, and drive competitive benefits.

Nowadays, customers can use cloud applications, platforms, infrastructure tools, and all kind of services to migrate and manage their applications. However, migration and management of applications to cloud computing must be done in a strategic and methodical way. Before moving to cloud, enterprises must take some steps to determine whether their applications can or will benefit from the migration to cloud. For example, enterprises must check and assess their applications to determine whether they are suitable for the migration process. They need to take into account of the required migration processes and management of the key application requirements including costs of migration, application redesign,

A. Samir, PhD student (✉)
Faculty of Computer Science, Libera Università di Bolzano, Bolzano, Italy
e-mail: aelgazazz@unibz.it

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_9

refactor, performance, lock-in risk, availability, security, privacy, policies, business value, technology issues, and much more.

There are some well-known cloud providers such as IBM, Google, Amazon, Oracle, Rackspace, and Microsoft. Thus, understanding essential requirements, knowing the benefits, understanding the drawbacks, and determining clear guidelines for cloud migration and management are needed tasks.

The rest of this chapter is structured as follows: Section 9.2 identifies the concepts and benefits associated with cloud computing. Section 9.3 depicts the motivations and considerations of cloud migration and management. Section 9.4 presents the challenges and research questions relating to both migration and management of cloud. Section 9.5 explores related work. After that, Sect. 9.6 presents the proposed approach that consists of two steps which are: (1) Specifying the high priority requirements of cloud migration and management. (2) Enhancing organizations knowledge to migrate their applications to the cloud based on a well-planned and well-understood strategy. To achieve step 2, CMMI is modified by adding a new element in such a process. Finally, Sect. 9.7 concludes the whole work and presents the future work.

9.2 Cloud Computing Concepts and Features

Cloud computing is a technology that allows users to access a large pool of virtualized resources that can be dynamically leased and released through the Internet in an on-demand fashion. Microsoft defines cloud computing as the ability to rent and deliver a pool of resources such as applications, platforms, servers, and disk space. IBM clarifies cloud computing as a solution in which all resources from software to hardware and network are provided rapidly to customers on demand. Rackspace specifies cloud computing as a broad range of deliverable services [2].

The emergence of cloud computing has made a tremendous impact on business enterprises. For example, cloud allows applications to scale up or down to meet the growing requirements of service loads. It rapidly allocates and de-allocates resources on demand. Moreover, providers do not need to invest in the infrastructure; they simply rent resources from cloud according to their requirements and pay for the usage. Cloud computing outsources service infrastructure to reduce business risks and to shrink maintenance expenses. It guarantees to provide high-quality services for customers. However, cloud computing still has significant challenges that need to be handled and understood by business communities to allow them adopt cloud and manage and migrate their applications. Some of these challenges are:

- Migrating systems and applications to cloud
- Choosing a strategic charging model
- Specifying service level agreements

- Considering costing models
- Identifying security type

There are many other challenges related to cloud migration and management; Sect. 9.4 provides more explanation on these.

9.3 Motivations and Considerations

Cloud computing environment improves application performance through increasing the provided resources to match current demands. This could happen by allowing the providers to monitor their usage, dynamically scale resources up or down, and pay based on what they used. Consequently, cloud providers can save the financial and operational costs of applications through migrating their applications to cloud and manage them by using a set of provided cloud tools.

Application migration is the process of redeploying an application on a new platform and infrastructure. The application can be migrated from an existing datacenter to public, private or hybrid cloud in order to avoid building, maintaining and managing infrastructure. In addition, migration can be adopted to provide capacity for steady workloads and to benefit from a pay-per-use expense model. Moreover, applications can be migrated from cloud to cloud or from cloud to datacenter depending on the business requirements such as reduced cost, obtaining new features, seeking to improve application overall performance, gathering a wide and a broad access, simplifying accessibility of application and services, and improving application quality from availability, flexibility, customizability, and reusability to security and privacy [3–5]. Furthermore, organizations are not only being able to transfer their applications to cloud but also they can migrate data or other business elements as well. As a result, migration to cloud is done to achieve the following:

- Business requirements: to consolidate services efficiently, save management and development cost, scale applications dynamically, attract more customers, do rapid upgrades, and specify high-ranked applications to be moved to cloud at reduced cost
- Security requirements: to protect data from disaster, store backups, virtualization, policy examination, service level agreements, compliance policies, access control and auditability
- Technology requirements: to specify cloud services (e.g., SaaS, PaaS, IaaS), estimate required storage, determine and optimize performance, ensure business availability, and use variety of software and storage

9.4 Challenges and Research Questions

Many organizations are trying to migrate their applications to a cloud infrastructure to increase resources, save cost, enhance performance, optimize applications, do better application management, rapidly process massive amounts of data, new technology and many more privileges provided by cloud. However, migrating applications to cloud pose some challenges and questions that are required to be tackled. From conducting literature review on current approaches [6–11], the following represents the research questions accompanied with applications migration and management:

- What are the main factors that business community should consider before moving to cloud?
- What are the criteria that business community must take into consideration to choose suitable cloud service?
- Regarding business and technical priority, how do business communities identify and rank applications and services that are best suited for moving to a cloud?
- What are the applications that are best suited for cloud deployment models?
- Which parts of the application will be migrated and which ones will locally reside on premises?
- What are the risks that may occur before and after migration?
- Is there any requirement for application modifications to migrate to cloud, and if there are, what are they?
- How will the applications benefit from migration?
- Which payment methods are best suited the migrated application?
- How will business community assess the tradeoff between costs and performance of the applications on local machines and on cloud environment?
- What kind of facilities do cloud providers offer, and which one will achieve the optimal process for the migration of application?
- How much effort is required to migrate applications to cloud?
- How do business communities secure their most important services once they are transferred to cloud?
- What are the security mechanisms that cloud providers support to secure, recover, retrieve, manage, and control businesses data?
- Can businesses reuse their existing resource management and configuration tools?
- How can businesses manage their contracts for network, software, and hardware?
- How can businesses allocate and de-allocate their resources on the cloud?
- Where do the applications get stored to?
- How can cloud customers get rid of provider lock-in?
- How can the dependencies of applications be specified before migration?
- Does a cloud provider support seamless integration with the communicated applications?
- Is there compatibility between organization, cloud's hardware and software?

- Does a cloud provider support simple customization and configuration tools?
- Does a cloud provider service level agreement include and document concerns about risk, security, privacy, policy, and performance?
- Does a provider offer application availability online and offline 24/7?
- Does a provider support multiple virtual machine images?
- Does a cloud provider allow seamless interface and wide access to the migrated application either from desktop computers, mobiles, or tablets?

Consequently, there are some concerns and obstacles that need to be considered before migrating applications to the cloud, e.g.:

- Provider lock-in: standards and technology should be considered before moving to a specific provider. Once the application is being hosted on the provider's site, it is hard to switch to another provider because the services, tools, and legislative may differ completely.
- Integration: it ensures seamless communications among integrated applications.
- Security: security plan should be wisely considered before moving customers' data to cloud such as the type of provided security, access control, authentication, recovery plan, knowing laws, regulation, data location and service level agreements, etc.
- Availability: it ensures that application, customer support, and technical aid are available 24/7. It addresses the mean time to repair, and the mean time between failures.
- Performance: the ability to know about server performance, determine type of hardware, identify compatibility, check services performance before and after migrating to cloud, define service level agreement, and know network bandwidth, number of users, buffer capacity, and the location where data will be stored.
- Payment plan: it specifies the type of payment method to avoid wasting resources.
- Portability: the ability of application to run on different platforms without facing compatibility challenges.
- Budget: it estimates the cost of migrating applications or data to cloud in order to check if the migration is worth it or not.

As depicted in this section, understanding and mitigating all of the previous questions and challenges will permit business communities to successfully migrate and manage their applications on cloud environment successfully without any failures.

9.5 Related Work

Cloud migration refers to the process of transferring an application as a whole or partially to cloud. However, after analyzing current works [12–23], the literature studies show that current organizations don't have a clear systematic approach or an

assessment plan which allows them to efficiently migrate and manage their applications, data, or services in cloud. In addition, current studies partially addressed the previous mentioned goals and challenges as shown in Sect. 9.4. Furthermore, current literature did not mention the high priority requirements of cloud migration and management. The following paragraphs explore recent approaches that targeted applications migration and management from various challenges and questions perspectives.

Lee and Kim [12] presented approaches for providing a high cloud services scalability. The authors presented two effective scalability assuring schemes which are service replication and migration. They proposed a set of steps to ensure service scalability management. However, only scalability have been considered in the application migration, and the rest of service qualities such as availability, performance, and security haven't been included. Armbrust et al. [13] only specified that elasticity is the significant feature in cloud as it transfers resources cost and management responsibility to provider. Falatah and Batarfi [14] introduced some considerations that solved scalability. Chieu et al. [15] presented a case study on the scalability and performance of web applications in a cloud. They introduced a scaling algorithm for automated provisioning of virtual resources.

Babar and Chauhan [16] studied the Hackystat framework to identify the steps to be considered before migrating applications to the cloud environment. The authors identified four key requirements for cloud migration and analyzed those requirements to gain an understanding of the changes that need to be made.

Palanisamy [17] specified ten key risks that should be considered in cloud; some of these risks are reducing governance, specifying data location, handling data ownership, and decreased monitoring. Sommerville et al. [18] demonstrated the potential benefits and risks associated with the migration of an IT system. They only focused on the cost, maintenance, and support. In addition, the authors mentioned the risks and benefits that organization might face during the migration to cloud. Thomas et al. [19] determined the fundamental risks that may occur from sharing infrastructure between mutual users through virtualization within a third-party, cloud service, and they proposed two ways to mitigate these risks. Ward et al. [20] discussed the impact of migration on cost and risks, and they provided an automated framework for seamless migration to cloud.

Akoramurthy and Priyaradhikadevi [21] analyzed both the migration path, the user's perspective, and the current tools of migration through proposing architecture that includes migration plan, recovery and consistency, transformation, and development. Edmonds et al. [22] proposed an open framework to relocate cloud services for cloud service developers and operators. Frey et al. [23] presented a simulation-based genetic algorithm that enhances cloud deployment options for supporting software cloud migration.

9.6 The Proposed Approach

Organizations need to assess their current applications before migrating them to cloud. The software migration processes must be applied correctly to get a better result in the form of a higher-quality software, more customer recruitment, satisfaction, and trust leading to more future revenues.

As depicted on the literature review, some researches provided framework, steps, or approaches to simplify migration; others conducted analysis as a guide process for migration. As a result, the high priority requirements of migration and management will be specified to aid the organizations to assess their current situation before migrating to cloud. In addition, CMMI will be modified to improve organizations knowledge to migrate their applications based on a well-planned and an understood strategy.

9.6.1 Migration and Management Requirements

This section provides high priority requirements of cloud migration and management which the vendor and provider should consider before transferring their applications and/or data to cloud services. The following requirements are proposed according to the literature study that has been conducted on 72 studies.

9.6.1.1 Security Requirements

Security is considered one of the most important and critical concerns for cloud computing migration and management. There are many requirements to achieve a high securable migration and management. These requirements are authentication (data, devices, users), authorization, access control, security auditing, code reviewing, malware detection, access log information, managing accounts, privacy, policy, allocate, de-allocate, encryption, harden of defensive methods, governance, threat management, service level agreement, security provisions, network security, data security, customization, configuration, application integration, etc.

9.6.1.2 Quality Requirements

Quality requirements play an important role in ensuring application and data: availability, scalability, reliability, fault tolerance, interoperability, adaptability, usability, reusability, flexibility, customizability, configurability, upgradability, agility, elasticity, cost-efficiency, mean time to repair, and mean time between failure and performance.

9.6.1.3 Operational and Technical Requirements

The following requirements are considered as the most crucial operational and technical aspects of migration and management: performance, low cost, deployment cost, centralized reporting, monitoring, virtualization management, customization, configuration, quality assurance and control, data and business processes, design requirements, availability, maintenance, interoperability, upgrades, capacity planning, data and application management, usage monitoring, billing methods and flexibility, service level agreement management, provisioning, load balancing, auditing tools reporting, testing, capacity planning, storage and processing control, seamless bug fixes and upgrades, patch management and process management, disaster recovery, backup, kind of management and monitoring tool, simple interface design, multitenancy, seamless integration, technical support, Cloudonomics, third party engagement, provider lock-in flexibility and transferability, scalability and redundancy capabilities, refactoring, etc.

9.6.1.4 Technology and Implementation Requirements

Before migrating applications, data, or services to cloud, there are technological aspects that should be specified. For example, application technology compatibility, interoperability, type and capacity of storage, kind of management and monitoring tools, suitable type of cloud services, specify the most proper cloud deployment, dedicated servers or virtual machine environment, test plan, performance, specify the priority of application, data, and services that are needed to be migrated to cloud first and which ones will reside locally on-premises, seamless integration, software and hardware optimization, fame of provider, software licensing, service level agreement, application portability, scalability, cost, and pricing model.

9.6.2 Enhancement Process

In the software industry, Capability Maturity Model Integration (CMMI) which is a process reference model is often used for guiding organizations to achieve software development. It refers to key process areas (KPAs) which identifies a cluster of related activities that, when performed collectively, achieve a set of goals. Each KPA has goals (Gs) which must be achieved, and each G contains several practices (Ps) for achieving the goal. A process area is satisfied when organization processes cover all the goals and practices for that process area. For example, Fig. 9.1 depicts a specific goal and its related specific practices for the Project Planning process area [24]. Thus, referring to the following figure, if the organization covers G1 and its P1.i of the project plan process area, then the project planning area is being achieved by organization.

- G 1 Establish Estimates
  - P 1.1 Estimate the Scope of the Project
  - P 1.2 Establish Work Product and Task Attributes
  - P 1.3 Define Project Life Cycle

Fig. 9.1. Project Planning process area goal and practices

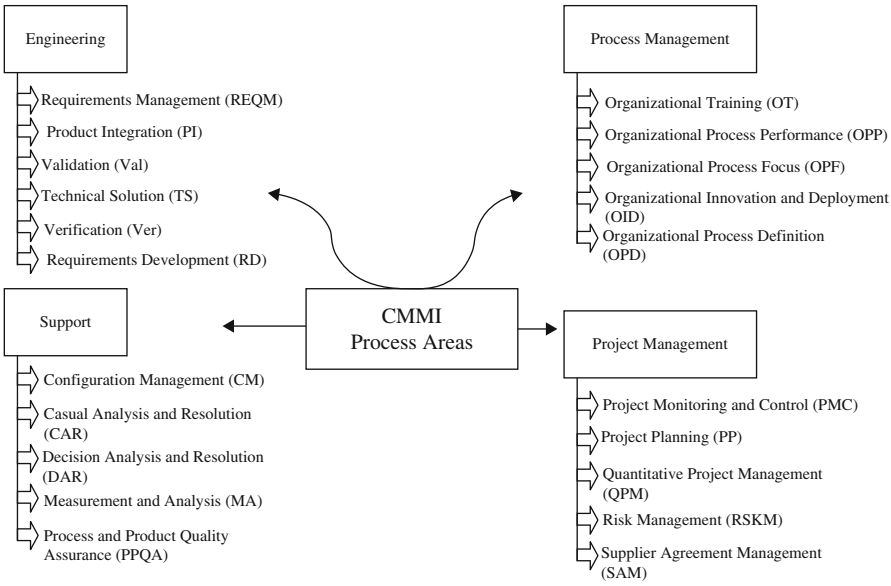


Fig. 9.2. CMMI process areas

The CMMI process areas can be grouped into four categories. Figure 9.2 depicts the interactions and links between them regardless of their defined level. The CMMI four categories are:

- Process management category
- Project management category
- Engineering category
- Support category

Each category contains a set of key process areas with their goals and practices. CMMI has five levels as shown in Fig. 9.3. Each process area is fallen into a specific CMMI level.

For example, CMMI level 2 contains the following key process areas [25]:

- Requirements management
- Project planning
- Project monitoring and control
- Supplier Agreement Management

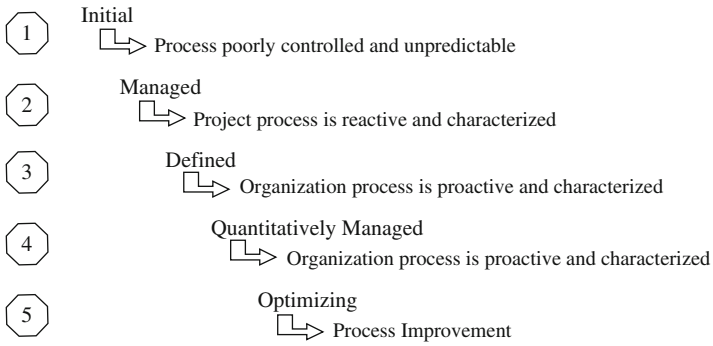


Fig. 9.3. CMMI levels

- Measurement and analysis
- Process and product quality assurance
- Configuration management

The KPAs can be considered as the most vital requirement for achieving a maturity level. Thus, to achieve a level 2 maturity, the KPAs for that level must be satisfied. The migration and management of applications to cloud requires an additional task to assess current applications, data, or services of organization before transferring them on cloud environment. This additional task is performing cloud migration and management assessment (CMMA) at the *managed level (CMMI level2)*. CMMA will help to check and evaluate the on-premise applications readiness and applicability for cloud infrastructure. In addition, it will aid in determining many requirements and answering multiple questions for the applications, services, and data being transferred on a cloud.

To assess applications, service, and data before moving to cloud, the enhancement process of cloud migration and management (CMMA) will be added to CMMI level 2 as a process area for the following reasons:

- All organization goals are achieved, requirements are managed, and all processes are planned, performed, measured, and controlled.
- Projects are performed and managed according to their documented plans because of the existence of retained practices.
- Requirements, processes, applications, and services are managed.
- Commitments are established and revised among relevant stakeholders, and products are reviewed and controlled with stakeholders.
- Costs are optimized and processes are improved.

Figure 9.4 illustrates and explains the main functionality of the modified CMMI level 2 process areas after adding the CMMA to the current 6 KPAs of CMMI level 2. Furthermore, Table 9.1 demonstrates the goals and practices of CMMA process area.

As depicted in Table 9.1, each goal represents a set of practices (requirements) that should be checked to satisfy the goal. Achieving the four goals of CMMA

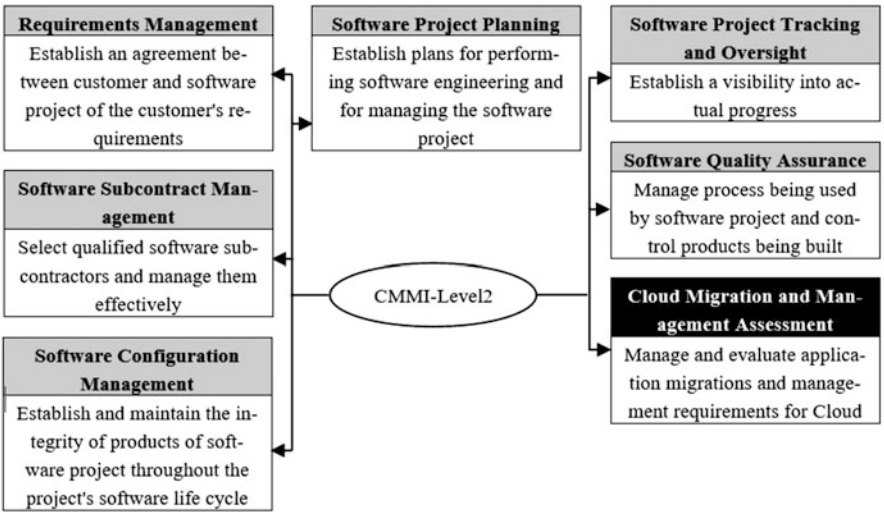


Fig. 9.4 Modified CMMI level 2

allows CMMA process area to be fulfilled by organizations. The following explains each practice of the CMMA process areas:

Assess Security

- Control access.
  - Specify the type of authentication and authorization.
  - Create access log file.
  - Manage users' accounts.
- Check defensive methods.
  - Specify defensive methods against attacks.
  - Create backups.
  - Duplicate data and distribute them.
  - Identify a recovery method.
  - Monitor and manage threats.
  - Use malware detection.
- Ensure seamless customization and configuration.
  - Provide tools to support customization and configuration.
  - Support flexible and simple user interface.
  - Support multitenant.

Table 9.1 CMMA process area goals and practices

Cloud migration and management assessment (CMMA) goals and practices	
Assess security	Control access
	Check defensive methods
	Ensure seamless customization and configuration
	Ensure application integration
	Document service level agreement
	Specify security auditing
	Ensure policy and privacy
Ensure quality	Ensure availability
	Document service level agreement
	Estimate performance
	Improve cost
	Assess modification and maintainability
	Manage multitenancy
	Control scalability and elasticity
Manage operational and technical requirements	Assess operation and management concerns
	Document service level agreement
	Assess operational costs
	Ensure flexibility of operations
	Identify recovery plan
	Specify hardware and software requirements
	Guarantee technical support
	Encourage simplicity and flexibility
	Specify test plan
Technology and implementation requirements	Specify application requirements
	Specify application, data, service priority
	Identify optimization plan
	Document service level agreement
	Specify cloud appropriateness
	Specify software license
	Specify cost
	Choose payment method

- Ensure application integration.
  - Allow on-premises applications to easily communicate with cloud applications.
  - Manage relationship dependency among applications.
- Document service level agreement.
  - Document the type of security services and the required standard which is provided by the service provider.

- Specify security auditing.
  - Check whether an application meet both the legal expectations of specified data protection and the organization's standards of achieving financial success against different security threats.
- Ensure policy and privacy.
  - Use encryption to protect data.
  - Specify strong authentication.
  - Identify legality and regulatory issues.
  - Robust separation between data.
  - Secure data in transit.

Ensure Quality

- Ensure availability:
  - Know mean time between failures.
  - Specify mean time to repair.
  - Ensure availability of application services.
  - Ensure the existence of technical support 24/7.
- Document service level agreement:
  - Document the availability of services (e.g., guarantees that the service will be available 99.99% during work days, 99.9% for nights/weekends).
- Estimate performance:
  - Ensure agility of applications.
  - Seamless customization and configuration that doesn't have a negative effect on performance.
- Improve cost:
  - Cloud's high-quality services shrink the spending on associated hardware, software, or licensing fees.
- Assess modification and maintainability:
  - Support reusability within applications.
  - Ensure that upgrades are not violating existing releases.
  - Allow application to run on different environments.
  - Allow the application to adapt to changes in requirements.
  - Adopt a good fault tolerance technique.
- Manage multitenancy:
  - Support multitenant in cloud applications.
  - Manage customers' accounts and separate their data efficiently.

- Control scalability and elasticity:
  - Support auto-scaling capacity.
  - Estimate cost per unit.
  - Adapt to workload changes.

Manage Operational and Technical Requirements

- Assess operation and management concerns.
  - Improve hardware and software performance.
  - Provide a suitable payment method for cloud services.
  - Support centralized reporting, virtualization process, and management.
  - Monitor hardware and software usage.
  - Allow refactoring, customization, and configuration.
- Document service level agreement.
  - Document the capability of provided services such as capacity, speed, performance, network, bandwidth, size, etc.
- Assess operational costs.
  - Estimate economics and financial requirements.
  - Specify a suitable payment method.
- Ensure flexibility of operations.
  - Support multitenancy.
- Identify recovery plan.
  - Support bug fixes and upgrades.
  - Ensure disaster recovery, backup. and data restoring.
- Specify hardware and software requirements.
  - Support scalability and redundancy capabilities.
- Guarantee technical support.
  - Availability of services.
- Encourage simplicity and flexibility.
  - Ensure seamless provider transferability.
  - Provide seamless integration between communicated applications.
  - Simple interface design.
- Specify test plan.
  - Estimate application performance before and after migration.
  - Ensure that load testing has been performed well, and all applications and operations are running without any errors in the new environment.

Technology and Implementation Requirements

- Specify application requirements.
  - Ensure compatibility, interoperability, integration, type, and capacity of hardware and software.
  - Specify suitable management and monitoring tools.
  - Identify location of the data (e.g., consistent with local legislation).
- Specify application, data, and service priority.
  - Specify which application, data, or service will reside on-premise and which one will be migrated to cloud.
  - Rank the infrastructures, platforms, and services which are planned to be moved to cloud.
- Identify optimization plan.
  - Check whether the application and hardware that are intended to be migrated have been fully optimized for cloud.
  - Explore the sensibility of optimizing current hardware infrastructure before moving to cloud.
- Document service level agreement.
  - Allow application and data portability.
  - Document all implementation and technology requirements.
- Specify cloud appropriateness.
  - Which cloud type (private, public, or hybrid) is best suited for organizations' needs. If the application has a sensitive data, then a private cloud is the best choice.
  - Specify whether dedicated servers or virtual machine environment are addressing organization requirements.
  - Know the reputation of provider.
- Specify software license.
  - Specify a licensing model.
  - Ensure that software license is compatible with cloud.
  - Ensure that software licenses have no restrictions on the maintained backup copies.
- Specify cost.
  - Estimate the cost of the used hardware and software.
- Choose payment method.
  - Specify a suitable pricing model.

Tables 9.2, 9.3, 9.4 and 9.5 concisely explain each practice of CMMA.

Table 9.2 CMMA practices

CMMA goals	CMMA practices	Explanation
Ensure quality	Ensure availability	Support application availability
	Document service level agreement	Guarantee the availability of application and document it within the contract
	Estimate performance	Ensure a seamless application performance
	Improve cost	Provide high-cloud quality services to reduce hardware and software cost
	Assess modification and maintainability	Ensure that the application enhances maintainability and simply adapts to requirement changes
	Manage multitenancy	Manage and separate users data
	Control scalability and elasticity	Manage application to scale up or down based on different needs

Table 9.3 CMMA practices

CMMA goals	CMMA practices	Explanation
Assess security	Control access	Control and restrict users access
	Check defensive methods	Specify preservative methods against attacks and threats
	Ensure seamless customization and configuration	Support simple and multitenant customization and configuration tool
	Ensure application integration	Allow simple integration among on-premise applications and cloud
	Document service level agreement	A contract that includes the type of provided services by providers
	Specify security auditing	Ensure that the application achieves the organization expected standards of protection
	Ensure policy and privacy	Specify policy and privacy standard for the application

9.7 Conclusions and Future Work

Cloud computing is on-demand access to a shared pool of computing resources. It helps consumers to reduce costs, decrease management responsibilities, and increase business agility.

This chapter discusses cloud migration and management of current approaches, benefits, concepts, and challenges. In addition, the high priority requirements for migrating and managing applications into cloud are provided and categorized into four major requirements. CMMI has been chosen to aid providers, and vendors assess the maturity of their organizations and applications to be migrated to cloud infrastructure. CMMI level 2 has been selected as organization goals, requirements, and processes are achieved, planned, managed, measured, and controlled. The process of enhancing CMMI level 2 is occurred through identifying four steps.

Table 9.4 CMMA practices

CMMA goals	CMMA practices	Explanation
Manage operational and technical requirements	Assess operation and management concerns	Evaluate hardware and software performance, manage refactoring, provide a suitable payment method and manage virtual machine images
	Document service level agreement	A contract should include and explicitly mention services capability
	Assess operational costs	Specify financial and economics different requirements
	Ensure flexibility of operations	Handle different user requirements and provide a clear customization and configuration methods to fit each user
	Identify recovery plan	Specify a recovery plan and do backups to save users data
	Specify hardware and software requirements	Support hardware scalability and make multiple copies of data to be protected
	Guarantee technical support	Provide technical support 24/7, live chat, data backup and expert people to support users. Describe approach to provide technical support to enable Agency migrate to cloud services including provisioning capabilities, accounting capabilities and billing capabilities
	Encourage simplicity and flexibility	Provide simple and flexible interface for softwares to be easily integrated
	Specify test plan	Test application before and after the migration to check whether it achieves the expected requirements

Table 9.5 CMMA practices

CMMA goals	CMMA practices	Explanation
Technology and implementation requirements	Specify application requirements	Achieve compatible, interoperable, integrated application and identify the location of data backup
	Specify application, data, service priority	Specify and rank the priority of existing applications, services or data to be migrated to cloud
	Identify optimization plan	Specify optimization steps for software before migrating to cloud
	Document service level agreement	A contract that documents technology requirements which is provided by provider
	Specify cloud appropriateness	Specify the necessity of migration to cloud
	Specify software license	Assess the compatibility of software license before migrating to cloud
	Specify cost	Estimate the total cost of the needed cloud hardware and software before migrating
	Choose payment method	Specify a pricing model based on the required performance

First, the challenges of cloud migration and management are identified. Second, the research questions of cloud migration and management are elicited from the existed literature studies. Third, the most important requirements are grouped with a newly added key process area under the name of Cloud Migration and Management Assessment (CMMA). Fourth, the goals and practices for CMMA process area are proposed and identified.

To conclude, the aim of this chapter is to aid providers and vendors migrating and managing their applications without facing future failures. The proposed approach allows providers and vendors to achieve a complete migration process with less migration cost and with high quality. This work can be extended through modeling relationships and dependencies between goals and practices. Evaluating the proposed approach to depict its effectiveness in managing and migrating applications to cloud.

References

1. Cearley DW (2010) Cloud computing: key initiative overview. Gartner Report linkedin.com/pulse/20141117105234-958990-25-definitions-of-cloud-computing
2. CliQr (2015) CliQr application migration and management. Gartner report. http://cdn2.hubspot.net/hub/194983/file-2528709477-pdf/docs/Application-Migration-and-Management-1214-WEB.pdf
3. Cisco (2010) Planning the migration of enterprise applications to the cloud. Cisco Report https://www.cisco.com/en/US/services/ps2961/ps10364/ps10370/ps11104/Migration_of_Enterprise_Apps_to_Cloud_White_Paper.pdf
4. Cloud Standards Customer Council (2013) Migrating applications to public cloud services: roadmap for success. Cloud-Council Report. http://www.cloud-council.org/deliverables/CSCC-Migrating-Applications-to-Public-Cloud-Services-Roadmap-for-Success.pdf
5. Vashishtha H et al. (2012) Migrating a legacy web-based document-analysis application to Hadoop and HBase: an experience report. *Migrating legacy applications: challenges in service oriented architecture and cloud computing environments. IGI Global*. 226–247
6. Brijesh D (2010) Assess enterprise applications for cloud migration. IBM Report. http://www.ibm.com/developerworks/cloud/library/cl-assessport/
7. Scott B (2014) Your guide for moving applications to cloud. EMC2 Report https://infocus.emc.com/scott-bils/your-guide-for-moving-applications-to-cloud/
8. Caroline D (2015) The cloud migration checklist: what to consider. Computer weekly report http://www.computerweekly.com/blog/Ahead-in-the-Clouds/The-cloud-migration-checklist-What-to-consider
9. Paliwal S (2014) Performance challenges in cloud computing.
10. Shrikant DB (2013) Cloud migration benefits and its challenges issue. Int J Comput Eng 1(8):40–45
11. Zia A, Khan MN (2012) Identifying key challenges in performance issues in cloud computing. Int J Mod Educ Comput Sci 4(10):59
12. Lee JY, Kim SD (2010) Software approaches to assuring high scalability in cloud computing. Proc 7th IEEE Int Conf, Shanghai China, 2010
13. Fox A et al. (2009) Above the clouds: a Berkeley view of cloud computing. Dept. Electrical Eng. and Computer Sciences, University of California, Berkeley, Rep. UCB/EECS. 28(13)
14. Falatah M, Batarfi OA (2014) Cloud scalability considerations. Int J Comp Sci Eng Surv 5(4):37

15. Chieu TC et al (2011) Scalability and performance of web applications in a compute cloud. Proc 8th IEEE Int Conf, Beijing China, 2011
16. Babar MA, Chauhan MA (2011) A tale of migration to cloud computing for sharing experiences and observations. Proc 2nd ACM Int Workshop on SECLOUD. 2011
17. Balaji P (2012) Top 10 risks in the cloud. Coalfire report https://www.coalfire.com/medialib/assets/PDFs/Perspectives/Coalfire-Top-10-Risks-in-the-Cloud.pdf
18. Sommerville I et al (2010) Cloud migration: a case study of migrating an enterprise IT system to IaaS. Proc 3rd IEEE Int Conf, Miami USA. 2010
19. Ristenpart T et al (2009) Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. Proc 16th ACM Int Conf, Chicago USA 2009
20. Ward C et al (2010) Workload migration into clouds challenges, experiences, opportunities. Proc 3rd IEEE Int Conf, Miami USA. 2010
21. Akoramurthy B, Priyaradhikadevi T (2015) A quantitative study on migration path from legacy system to contemporary systems. Int J Res Comp Appl Robot 3(3):9
22. Edmonds A et al (2013) FluidCloud: an open framework for relocation of cloud services. Proc 5th USENIX workshop on hot topics in cloud computing. 2013
23. Frey S et al (2013) Search-based genetic optimization for deployment and reconfiguration of software in the cloud. Proc 13th IEEE Int Conf, Piscataway USA. 2013
24. CMMI. Key process areas http://www.tutorialspoint.com/cmmi/cmmi-process-areas.htm, Accessed 6 July 2016
25. Software Engineering Institute. http://sei.cmu.edu/cmmi/. Accessed 6 July 2016

Chapter 10
A Novel Approach to Modelling Distributed Systems: Using Large-Scale Multi-agent Systems

Bogdan Okreša Đurić

10.1 Why It Is Important to Consider Organisation

Recent developments introduced by the modern technologically advanced era have led to increased use of virtual (software) agents as opposed to using real-life agents, i.e. people in various scenarios. People have always been attracted to the idea of organisation. Groups of people were formed ever since *Homo sapiens*, and our ancestors, discovered benefits of socialisation, either in a planned manner or motivated by a common need of some kind, e.g. shelter, defence and hunger. Such organisations consisted of anything from only a couple of individuals (e.g. ancestors of the modern *Homo sapiens* in search of fulfilment of the mentioned needs) to as many as needed for a great army of the twentieth century.

Organisation applicable to the mentioned examples can be perceived as having one main function – overcoming various limitations of individual agents [48, 62, 63]. These limitations have several aspects, e.g. temporal (one agent has temporally limited availability), functional (some actions demand simultaneous effort, and an individual agent may not be capable of offering such a functionality), etc. Speaking of organisation amongst agents in a multi-agent system (MAS) implies a MAS comprising intelligent agents that can interact with each other and the environment, reason, act and react upon their perceived environment, communicate with each other, observe changes in the system, etc. It has been a prevalent thought in MAS-related studies that such agents can successfully serve as models of real-life situations and real-life people. Likewise, it is considered that human organisations, and principles of human organisation, can be successfully modelled using interactive intelligent agents. Although modelling agents conforming to their real-life

B. Okreša Đurić (✉)
Artificial Intelligence Laboratory, Faculty of Organization and Informatics, University of Zagreb, Pavlinska 2, 42000 Varazdin, Croatia
e-mail: dokresa@foi.hr

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_10

counterparts can be a tough goal to achieve, it is rewarding in the long run, since various experiments can be conducted within lower budget capacity, shorter time periods and with greatly increased reproducibility.

Organisation is not the only way to structurally model a system of agents. Another popular way of structured interaction and functioning of a MAS is swarm intelligence. While organisation model is derived directly from the well-known and researched concept of human organisation, swarm implies close ties to swarms of insects or similar forms of life. Each of these approaches to building MAS (or even large-scale MAS (LSMAS)) presents their researchers with different features of the resulting system and is thus more suitable for a specific application. Organisational modelling on one hand is about defined structure in the system followed by structured communication protocols in terms of hierarchy or possible ways of communication flows. Furthermore, organisational ideas can be known by individual agents directly, or they can be imposed on the given system, with agents being aware of the enforced elements of organisation, or the included agents can be ignorant of the system's organisational constraints, etc. Swarm [18, 41], on the other hand, is all about self-organisation and emerging organisation. Imagine a swarm of bees [8], a school of fish [40] or a colony of ants – there is no specific entity that would govern their behaviour, rather they act as part of a group which benefits from their individual behaviour. Structure therefore emerges from the group, based on performance of individual agents. The main difference, when observing organisation, is in the way organisation elements are formed – starting at the level of an individual agent (bottom-up), or from the level of the whole given system (top-down).

Each of these approaches is beneficial to specific scenarios, depending on many variables, including the type of environment of the modelled agents, the type of agents and their abilities, the main goal of the system, etc. Therefore, an extremely biased approach may not be the best way to model a system [13, 43]. Furthermore, it is rather easy imagining an example where both approaches intertwine, e.g. interaction of the swarm creates a certain organisation-like structure which is propagated further and strengthened until the swarm, probably provoked by another need of theirs, in a non-unison way, decides that organisational dynamics is in order, and the structure changes, if only for a small amount.

Backed by a notable development of computer power in the past few decades, rising popularity of agent-based structures and agent-aided distributed computing can be attributed to the rising complexity of software problems and, for example, the use of computing for conducting research on a global scale. From a different perspective, agent-based distributed computing is very beneficial to, and benefits from, the rising number of individual computer-imbued things a person can possess. For smart cars, smart bicycles, smart phones, smart cups, smart homes and smart cities, the potential for connecting all the existing pieces of software that are capable of connecting to, for example, the Internet is obvious. It may be seen as most advantageous to observe these pieces of software residing in many household things as agents, and the Internet, or a local network comprising these agents, as a MAS. Furthermore, it is argued that the efficiency of such systems is raised using

structured organisation, imposed upon the system, since it benefits more from the existent number of agents, their possibilities, and their focus on achieving a joint goal [22, 25, 29].

The mentioned scenario forms a basis for the Internet of Things (IoT), or, in an even more general case and of larger scale, the Internet of Everything (IoE). While IoT is clearly concerned with things and objects that are able to interact and cooperate with each other to reach their common goals [48], thus creating a rather clear possibility of being abstracted as MAS, IoE covers a much wider domain comprising people, processes, data and things working together to make appropriate and beneficial connections, more so than ever before [48].

Examples of IoE or IoT paradigms are applicable to various domains. Recent studies at the Artificial Intelligence Laboratory of Faculty of Organisation and Informatics at the University of Zagreb (AI Lab @ FOI) studied smart cities and agents in a massively multiplayer online role-playing game (MMORPG). When thinking in terms of smart cities or MMORPGs as application domains of LSMAS, it is favourable to think about organisation of included agents. Furthermore, it is advantageous to import various features of a human organisation into a system of agents.

Smart planning is a crucial element in planning and realisation of a project. There are several modelling methods, the most popular of which may be the UML notation, but only few are designed especially for MAS, let alone LSMAS. Organisational modelling of an LSMAS may be considered as planning a system of agents.

An organisational metamodel for modelling of LSMAS is being developed at the aforementioned AI Lab. The general idea and goal of this research is to develop an extensive model that would encompass several different organisational models and structures (e.g. horizontal vs. vertical organisation). Such a model will utilise a clear visual representation of the modelled concepts and will make it easier to plan an LSMAS, since it will incorporate elements of various perspectives of organisational modelling, e.g. organisational culture, strategy or organisational change. Furthermore, it will be possible, when the model is finished, for the user to generate a programming code skeleton, based on the built model.

The approach just described will make it possible for the user to build a model where most of the elements of the future system are specified. This step of creating an LSMAS is of great significance, wherefore the approach that favours change and alteration is most welcome, and this metamodel will offer one such approach. Visual design of a model will surely make it easier for the user to review the built model and to incorporate changes identified when evaluating the model built. Usefulness of code-generating part of the metamodel is obvious with respect to definitions built in the model. Since the code-generating process is automated and based on the built model, the outcome is bound to be represented by the said model built by the user. The metamodel is envisioned as a rather abstract view of the system though. Therefore, the generated computer code will represent only basics, and the programmer is expected to fill in all the needed detail.

The role of an organisational (meta)model is therefore obvious in planning and development stages of a distributed computing software project that relies on agents and their interaction.

The rest of the chapter is represented as follows. The rest of this section contains further details about MAS, emphasising roles of IoE and MMORPG and how they are related to LSMAS. Some basic organisational elements observable, and beneficial to, IoE and MMORPG will be noted as well. A brief overview of organisation of MAS and, more specifically, LSMAS will be given in Sect. 10.2. Section 10.3 covers two distinct use cases for the proposed organisational metamodel, repercussions of which, along with feedback, are discussed in Sect. 10.4. Brief overview and guidelines for further research are covered in Sect. 10.5.

10.1.1 About the Internet of Everything and Massively Multiplayer Online Games

It was mentioned earlier that, in their most basic form, an agent is a software entity surrounded by an environment. An interactive intelligent agent can perceive this environment of theirs and act upon it [45]. Such an agent can be considered a virtual representation of a human in a group or a system.

Although multi-agent systems represent an area where a lot of research has been done already, their larger counterpart, LSMAS, has had some research done only recently. Probably the most well-known concepts where LSMAS may be applied are the application areas of IoT [4, 55] and IoE. Even though IoT and IoE are used almost synonymously, there is a slight difference between the two, as nicely put by Cisco [36]:

> *In terms of phases or eras, Cisco believes that many organizations are currently experiencing the Internet of Things (IoT), the networked connection of physical objects. As things add capabilities like context awareness, increased processing power, and energy independence, and as more people and new types of information are connected, IoT becomes an Internet of Everything – a network of networks where billions or even trillions of connections create unprecedented opportunities as well as new risks.*

IoE seems to be the inevitable future of distributed computing and the core idea of distributed systems. Furthermore, some indications of a novel concept of the Internet of Agents appeared recently, e.g. [61]. A notable IoE area of application is smart cities [51–53, 57]. When thinking about a city filled with agent-controlled elements (e.g. cars, traffic lights, parking lots, buildings and homes, etc.), it may do well to think about organisation features amongst all the included agents. As opposed to swarming agents and emerging organisation traits based on behaviour of agents, agents in a city would be demanded from and per se inclined to fulfil a given task in an optimal amount of time, using the least resources and in the safest way possible. Such a task undertaken by every of thousands of agents would greatly benefit from features mirrored from human organisations, such as communication

protocols, rules of conduct, and similar. IoE, abstracted by LSMAS in a way similar to the one described may be applied to other domains, e.g. smart power grids, smart health, smart transport, smart buildings, etc., where some of the elements may be considered as sub-elements of, for example, smart cities.

IoE is a rapidly developing area that can be abstracted by LSMAS. Another example of great interest is domain of massively multiplayer online games (MMOGs). An MMOG is a computer game meant for a great number of players simultaneously playing the game online, often engaged in interaction with each other. MMORPG is a special kind of an MMOG that allows players to take control of their avatar (in-game character of the player) and interact with usually vast virtual worlds where many automated (nonplayer) characters and other players' characters reside [49]. Such games represent proper LSMAS environments: there are numerous agents (some controlled by players, most acting independently) with many opportunities to interact (e.g. trading, combat, training, pillaging, cooperation, communication, delegation, etc.) and a big world to explore (sometimes consisting only of towns, areas and cities, but some expand to planets and solar systems). In order to succeed, players often have to cooperate, i.e. join in smaller or larger groups, where they have to exercise real-life-like interaction with others, including choosing a leader, following orders or planning an attack. Such worlds are obviously very interesting grounds for training and experimenting with agents in an LSMAS.

10.2 Overview of Models for Organising Agents and State of the Art

As mentioned by several studies, only some of which are [10, 25, 29, 31] imposing organisation features on an LSMAS may bring more benefit to the system, than using the agent-centred paradigm. Therefore, it is interesting to talk about organising systems of agents. As mentioned earlier, copying elements of human organisations and applying them to artificial agents is the prevalent method of developing organisational features for systems of agents.

In this section an overview of organisational aspects meant for systems of agents is followed by a modern view of organisational modelling, needed for modern systems of large scale, as proposed in recent studies on LSMAS and the IoE.

10.2.1 Existing Models for Organisation of MAS

Organisational models and frameworks for organisational modelling of MAS have usually concentrated on structural features of an organisation. Organisational structure, as a primary feature being modelled, is often accompanied by concepts used

Table 10.1 Models and frameworks for organising MAS and their respective dimensions, according to the containing concepts, as described in [14]

Organisational model	Dimensions
AGR	Structure, interaction, agents
TÆMS	Functions, processes, environment
MOISE+	Structure, functions, norms
ISLANDER	Structure, norms, interaction
OperA	Structure, functions, norms, interaction
AUML	Structure, functions, interaction, environment
NOSHAPE MAS	Structure, dynamics, agents
MACODO	Structure, dynamics, agents

for modelling functional aspects of an organisation and concepts which aid in modelling agent interaction within a MAS. Other organisational features, such as norms or the environment in which agents are situated, are rather scarce in the MAS organisational models and frameworks developed to date.

This section covers an overview of some of the more popular means of modelling organisation of MAS, as shown in Table 10.1, along with their most significant dimensions (features) as described by Coutinho et al. in [14].

Models mentioned in Table 10.1 are further described by their authors in their respective papers. Some basic pieces of information about concepts used by each of the stated models are laid out further in this chapter. Such an overview is an introduction to perspectives of LSMAS organisational modelling.

AGR The *agent*/*group*/*role* model, or AGR model in short, was developed by Ferber et al. [22] and is also known as *Aalaadin* model. The three basic concepts included are meant for modelling individual agents, groups of agents and agent roles. The concept of agent within AGR conforms to features of agents mentioned earlier – it denotes individuals capable of interacting and communicating that can range within both extremes of reactivity and intelligence. The main trait of these agents, as the model is not concerned with their internal architecture, is that an agent plays roles and belongs to groups. A group consists of many agents that share a common interest or a characteristic. Thus, a group can be used for creating organisational segments and functional or structural parts of an organisation.

TÆMS As a framework developed by Decker, presented in [16], originally intended for modelling of complex computational tasks, *Task Analysis*, *Environment Modelling*, *and Simulation* framework can be used with MAS as well. The most prominent feature of TÆMS related to MAS is layered description of environments (not exclusively of the same meaning as environment in MAS). Closely connected to concepts describing environment are concepts for statements about tasks and task groups. Three layers described in [16] (objective, subjective and generative) are defined as levels of environmental and task characteristics model. It is interesting to note that TÆMS models an agent as a locus of belief and action.

MOISE+ Building on *Model of Organisation for multI-agent SystEms* (MOISE) and *Aalaadin*, both organisation-centred models, MOISE+ comprises concepts for structural, functional and deontic specification of organisation in a MAS. Although direct modelling of agents is not possible, MOISE+ depends on modelling roles, relations amongst them and groups. [30] Roles represent constraints individual agents must follow when playing a specific role. Possible roles an agent can play depend on the roles the given agent is playing already. Upon accepting to play a role, the given individual agent is added to a group playing that specific role. Another point of interest in MOISE+ is functional specification, wherein goals are structured in plans and grouped in missions. It should be mentioned that notation for sequential, parallel, and choice-based plans is present.

ISLANDER Seemingly situated slightly off of the centerpoint of MAS modelling, ISLANDER is a language for textual specification of electronic institutions [20]. Main parts of the language are used for specifying performative structure, scenes that make up the said structure and normative rules. Scenes serve as meeting points for agents communicating according to well-defined protocols. Roles again represent specific constraints over individual agents, along with specifying their possible actions (e.g. communication protocol). Normative rules set up agent actions that have consequences of some gravity.

OperA This framework developed by Dignum presented in [17] is primarily focused on describing system at a conceptual level. Therefore, the developed concepts are mainly used to define structure and global behaviour of the model, including, for example, organisational characteristics, while individual agents that populate the said model are modelled separately and independently of their internal design. Such a feature is achieved using three components: organisational model, social model and interaction model. The organisational model encompasses concepts of roles and interactions, the social model populates the defined organisational structure with agents playing roles, and the interaction model is built using interaction between agents.

AUML During the year 2001, an effort was made, described by Van Dyke Parunak and Odell in [19], in order to enrich *Unified Modelling Language* (UML) with concepts useful for agent-based systems (i.e. MAS). Concepts that were identified as most useful are swimlanes, class diagram, sequence diagram and activity graph. Swimlanes were proposed as representation of groups of roles, along with role instantiation. Class diagrams were used to define roles and their relationships, similar to swimlanes enhanced by cardinality constraints. Sequence diagrams were used to describe possible interaction amongst various agent roles. In the end, interaction of groups and group-level dependencies, where these groups can be modelled as agents, was shown using an activity graph.

NOSHAPE MAS The main purpose of this novel organisational model is to be the most general one of those mentioned here. NOSHAPE recognises three levels of abstraction: universe, world and organisation. Using concepts of holarchy and hierarchy, Abbas [1, 2] thinks of agents as individuals or groups depending on

the perspective: bottom-up perspective sees a group, while top-down perspective is concerned with agents as individuals. Therefore, levels of abstraction consist of several individuals of lower level abstractions (e.g. a universe comprises an infinite number of worlds). Each of these agents is situated in an environment and can interact with each other. Naturally, the concept of roles is existent as well. An interesting observation is static roles, such as Global Supervisor and Local Supervisors – roles that are concerned with organisational structure or organisational behaviour.

MACODO This organisational model used for describing dynamic organisations is a part of an integrated approach called *Middleware Architecture for Context-driven Dynamic agent Organisations* (MACODO) [58, 59]. The main feature of this model is that agents are modelled separated from their life cycle, thus making it easier to understand, and model, how changes in the system, or changes in the environment, affect dynamic organisations, i.e. agents. Agents are uniquely identified within the system and have their capabilities grouped into sets called roles.

This brief overview of some of the better-known organisational models or framework indicates that said models, as shown in Table 10.1, usually comprise concepts describing organisational structure of a system of agents (e.g. groups of agents), interaction of agents (e.g. communication protocols), normative restrictions (e.g. norms in context of constraints over agents and their capabilities or rules of conduct), functional features of an organisation (e.g. capabilities of agents), etc. All the mentioned models, except the most recent one, NOSHAPE MAS, are concerned with MAS in general, without mentioning LSMAS in particular. Only NOSHAPE MAS mentions several levels of abstraction and thus potential for a large-scale organisation.

10.2.2 Recent Advancements in LSMAS Organisational Models

As was mentioned earlier in this chapter, multi-agent systems of large scale have been recently shown as applied to the Internet of Things, or the Internet of Everything. One such example is coming from the medical area, specifically distributed worldwide healthcare applications, as described by Bui and Zorzi in [11]. The mentioned authors strive to create a communication framework for agents included in the system. This permits argument about communication methods of agents within LSMAS and requirements of that particular element of organisation.

When speaking of distributed systems consisting of autonomous agents, Scheutz noted in his 2010 research [50] that MAS did not have enough flexibility at the time, nor were they supportive enough, for sophisticated large-scale intelligence applications. The solution the mentioned author proposed was in synergy of MAS and system of single agents.

A rather long time ago in context of LSMAS, in 2002, McCauley and Franklin described an LSMAS used in US navy personnel distribution [35]. The described system looked after the needs of US navy entities taking care of, for example, their needs, state of the system, scheduled personnel changes, etc. Three main classes of agents existed: sailor agents, command agents and navy agents. Most of the communication and interaction took place between sailor and command agents (they negotiate available positions, sailor interests, etc.), while the navy agent acts as an overseer.

On the other side, and published more recently, research was done by Schatten that takes into account the interdisciplinary potential of LSMAS research [46]. The mentioned author uses complex analytical method (cro. kompleksna analitička metoda, KAM) to conduct self-organisation in MAS. In general, KAM is used to analyse organisations and propose organisational model that is new and optimised. Using KAM, and adapting it to MAS, is enhanced using the fractal principle (e.g. every agent is considered an organisational unit, but a group of agents that collaborate and have a common goal are considered an organisational unit as well). Such an approach is rather similar to ideas of holons and holarchy [3].

Another proposal intended to create an easier way to work with agents in LSMAS is described by Boulaire et al. in [9]. The mentioned authors propose an approach of dynamic agent composition that is intended to break agents into atomic units (i.e. parts) that together form a complete agent, and the whole system, and are combined at runtime. The three entities that form an agent are an asset, behaviours and data. This novel approach to building agent-based models (ABMs) aims to extend ABMs with underlying networked structure, thus allowing users to develop new elements of a system and add them to an existing system, without the need to access or modify previously written code.

Further research done by Schatten as elaborated in [47] is even more pertinent to large scale of MAS, and foundations are set for an ontology comprising concepts of organisational modelling applicable to the domain of LSMAS. This approach, of creating an initial ontology for modelling complex systems, is deemed necessary by the mentioned author, since it would allow for definition of formal semantics of the modelled systems. Furthermore, some conceptual foundations are laid for an LSMAS framework.

This research done by Schatten [47] provides several perspectives of organisational modelling that are recognised as crucial in future development of LSMAS, i.e. organisational models of LSMAS. Some of the concepts of such an ontology of organisational design methods are detailed in the chapter as well. The said perspectives are defined by Schatten [47] as follows:

- Organisational structure defines the decision and information flows of an organisation.
- Organisational culture defines important intangible aspects of an organisation including knowledge, norms, reward systems, language and similar.
- Strategy defines the long-term objectives of an organisation, action plans for their realisation as well as tools on how to measure success.

- Processes define the activities and procedures of an organisation.
- Individual agents define the most important asset of any organisation – the individuals actually performing the work.
- Organisational dynamics define organisational changes including reorganisation of any of the above-mentioned components.
- Context and interorganisational aspects define organisational behaviour towards its environment including strategic alliances, joint ventures, mergers, splits, spinouts and similar.

Some of the most recent studies [5–7, 15, 21, 24, 26–28, 32, 34, 36, 42, 44, 54, 56, 60] provide further incentive to think of research on systems of agents, especially those of large scale, as important.

A clear direction of thought is recognisable in some of the models mentioned in Sect. 10.2, as the following can be observed:

- Many of the mentioned models think of MAS and LSMAS on a number of levels of abstraction (since LSMAS may comprise thousands of individual agents, it may seem like a natural way of viewing such systems).
- Somewhat of a leitmotif is use of roles for introducing constraints or a set of features for individual agents.
- Grouping agents by roles is used often.
- It is curious that only the most recent studies think of dynamically changing systems.

10.3 The Metamodel and Examples

The organisational metamodel that is used in the following examples is being developed [37, 49] based on recent studies mostly presented in this chapter. The finished metamodel will be based on an ontology being developed by the author of this chapter [38, 39]. An overview of the metamodel at this very early stage of development is given in this section, followed by two examples of its use.

The following examples shall be used to demonstrate the early working version of the organisational metamodel for LSMAS being developed. The main idea of this metamodel is to adhere to the seven perspectives of organisational modelling of LSMAS mentioned by Schatten in [47] that are described informally in Sect. 10.2.2 of this chapter. Therefore, the individual agents are regarded as organisational units. Furthermore, an organisational unit can comprise infinite number of organisational units. What is even more interesting, such a relation will be used for even more organisational entities of the system being modelled, e.g. goals, tasks, etc. The metamodel in its final version is expected to include, amongst others, concepts related to organisational change and, the most complex element, organisational culture.

At the moment, the metamodel can be used for modelling the following concepts of an LSMAS: organisational unit, role, goal/objective, process, knowledge artefact

(KnArt) and several properties of these concepts, e.g. inclusion, possible roles of an organisational unit, command flow, etc. These organisational concepts have been identified upon analysis of an ontology containing selected organisational concepts used in organisational modelling of LSMAS [38, 39]. The ontology is a work-in-progress as well and is being built based on standard practices of organisational modelling of human organisations, but is clearly directed towards LSMAS. The aim of the finished metamodel is to comprise LSMAS organisational concepts identified based on the mentioned ontology and on the related research, all of which shall be in accordance with the modern features of LSMAS organisational modelling [47]. Distinction of the proposed metamodel, when compared with existent LSMAS organisational models, will be visible through elements such as included concepts for modelling interorganisational dynamics, and a "zoomable" approach, where many of the included concepts will be observable on various levels of abstraction, to name a few.

An organisational unit (see Fig. 10.1a) is the basic element which represents an individual agent. Alternately, an organisational unit can represent a type of agent (see difference between approaches in Sects. 10.3.1 and 10.3.2). Every organisational unit can access an individual knowledge artefact (Fig. 10.1b), and it can play any number of roles. Furthermore, an organisational unit can be a part of another organisational unit (Fig. 10.1d), thus creating a group, or it can answer to another organisational unit (Fig. 10.1c). Organisational units are not supposed to be detailed any further, since the emphasis of the model is on organisational and not individual modelling.

Roles are modelled with the idea of constraints in mind. Every role (Fig. 10.2a) has some dedicated actions which become available to an agent that plays it. Furthermore, roles have basic properties similar to those of organisational units: hierarchical command flow (Fig. 10.2d), grouping relation and access to organisational knowledge artefacts (Fig. 10.2b). As opposed to an organisational unit, a role can combine its available actions in a process that can be used to achieve a certain goal (Fig. 10.2f). Every role can have a specific goal (Fig. 10.2c) that can hierarchically consist of subgoals. Finally, a role can have a generic relationship with another role (Fig. 10.2e). Such a property allows for the developer to specify the connection they need. A role is thus somewhat of a central concept in modelling a system. Certainly, the approach depends on the will of the developer and purpose of the model.

A knowledge artefact contains a piece of knowledge of the system. Modelled as abstract representations of knowledge, knowledge artefacts are designed to be detailed by the developer once the system development starts after the modelling phase. Knowledge artefacts (KnArts) are divided into Individual and Organisational KnArts. Individual KnArts (Fig. 10.1b) contain knowledge of importance to individual agents. Organisational KnArts (Fig. 10.2b), on the other hand, represent pieces of knowledge pertaining to the organisational aspects of the system and are, by default, accessible to roles only.

Goals, i.e. objectives, are modelled to contain specific basic information about the given goal and what is needed for the goal to be fulfilled. Therefore, every goal

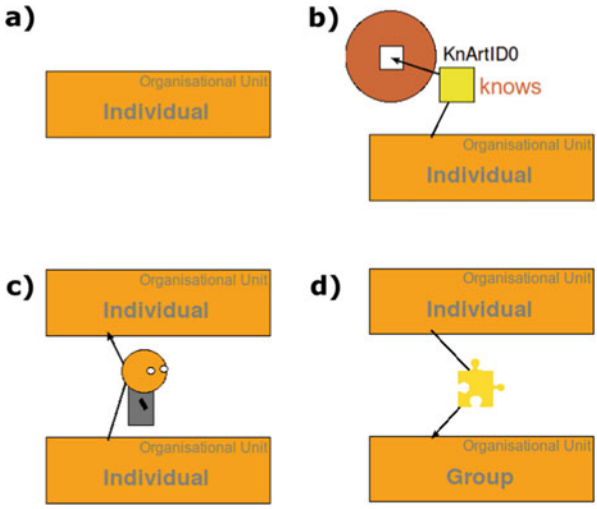


Fig. 10.1 Visual representation of the organisational unit concept

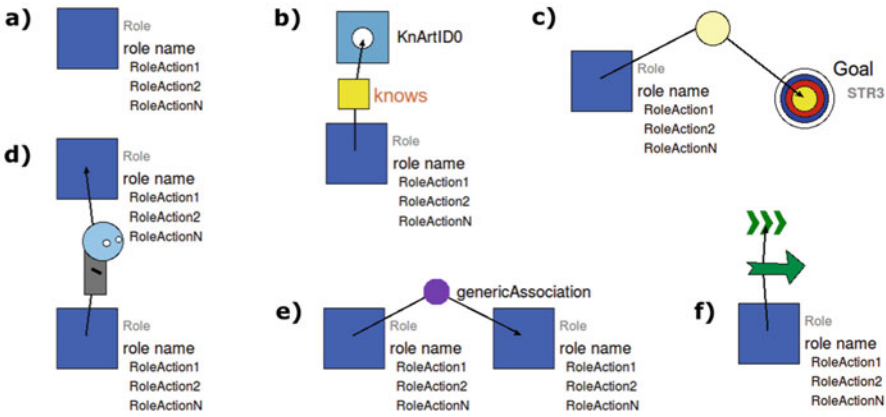


Fig. 10.2 Visual representation of the role concept

can have its respective measurement and reward values (Fig. 10.3). These are written in the way most apt for the development process of the system. Every goal may be a part of another goal concept, thus creating a hierarchy and subgoals.

A process concept is abstracted as a concept that can be enacted by a role and that has a certain goal concept for its outcome. Therefore, the outcome can be achieved using the designated process. A process can be used to achieve a certain subgoal as well, thus being useful in fulfilling complex goal concepts. Ideally, a process available to a specific role consists of actions available to that particular role.

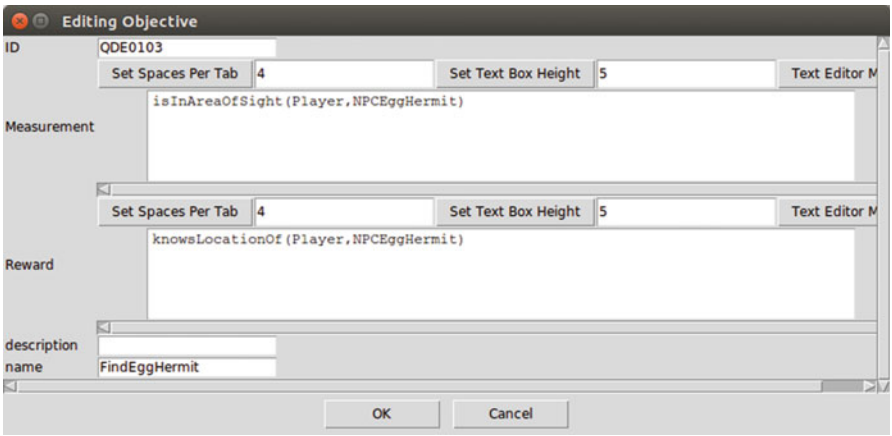


Fig. 10.3 Details of a goal concept are described using predicate logic statements

Properties included in the metamodel will not be detailed here, as their basic information was provided above, when other concepts were described in more detail.

10.3.1 A Simple Example of the RecipeWorld

The RecipeWorld was developed by Fontana and Terna as an "agent-based model that simulates the emergence of networks out of a decentralized autonomous interaction" [23]. This model is built using three basic elements: recipes, orders and agents. Recipes contain a custom number of steps that have to be taken if an objective is to be achieved. Orders are specific objectives that have to be fulfilled. Every order has some technical information and auxiliary data. Finally, agents are solving given problems, by completing steps defined by a recipe. Technical details (e.g. recipe structure) can be found in [23]. As noted by the mentioned authors, one of the goals of this agent-based model is to generate network based on activity of agents, instead of making agents generate a network a priori.

Typical application example of the described model is that of production. Several factories have to produce all the generated orders, thus creating a social network that can be analysed for specific insight into the production process. Additional constraints are introduced into the system (e.g. a specific factory can only work on a specific element of a recipe). Each factory of this system and each order to be produced are represented as agents. As the system run, production started, and the orders move around the system to factories that can produce the needed recipe part, and thus a network is generated.

In this example, individual structure or functioning of an agent is not of great concern, i.e. an agent will be modelled almost as a black box, giving the system

designer freedom to develop the agent in any way they want. Each agent will be given certain constraints though. First of all, roles will be used to determine if an agent is a factory, or an order. This way, a role will contain a set of constraints, and the agent playing the given role will have to act accordingly. Furthermore, knowledge of a recipe of the order will be modelled as a knowledge artefact. This way, the abstract concept of a knowledge artefact can be realised by the system developer in a way they desire (e.g. a rule language like RIF or SWRL). In order to utilise and simplify communication of agents involved in the system, a knowledge artefact specifying ontology of communication concepts will be accessible to all the roles of the system. Every role in the system will be related to a couple of needed functions or processes as well. This way, the whole process of organisational design will be moved to a more abstract layer, as opposed to working with individual agents. What is more, this type of declaration allows for computer code generation of the basic elements of the modelled system. The described process is detailed as follows.

One version of a situation modelled as described is shown in Fig. 10.4. Individual agents are represented using orange rectangles. An individual agent can play any of the connected roles represented as blue squares. Furthermore, as described above, every agent has individual knowledge of their recipe parts, or services they provide, based on the designated role they will be performing. This formulation presumes that individual agents performing different roles will be basically different. Every role has several actions that are clearly named and will most likely be translated into code. As mentioned above, both roles have access to a kind of a knowledge repository, a knowledge artefact of organisational concern, which stores the domain ontology. Each role has its respective main goal (i.e. objective) which is decomposed further. Subgoals are not defined as goals of the given role, but are achieved by processes available to a specific role. Two separate processes are available to the factory role, and their result should be fulfilling the goals they are connected to.

Model presented in Fig. 10.4 may yield programming code as follows. It is worth noting that the metamodel is a work-in-progress, and code generation is one of the features not yet implemented. Therefore, only a possible suggestion based on the model is shown. The programming code shown in the box below is based on Python and Smart Python multi-Agent Development Environment (SPADE), where agents are instances of an agent class and their behaviours are instances of behaviour classes covering various types of behaviours.

```
class AgentOrder(spade.Agent.Agent):
  class SearchForFactories(): [...]
  class CheckFactoryAvailability(): [...]
  class WaitForFactoryAnswer(): [...]
class StartProduction(): [...]
class FinishProduction(): [...]
def initialise ():
```

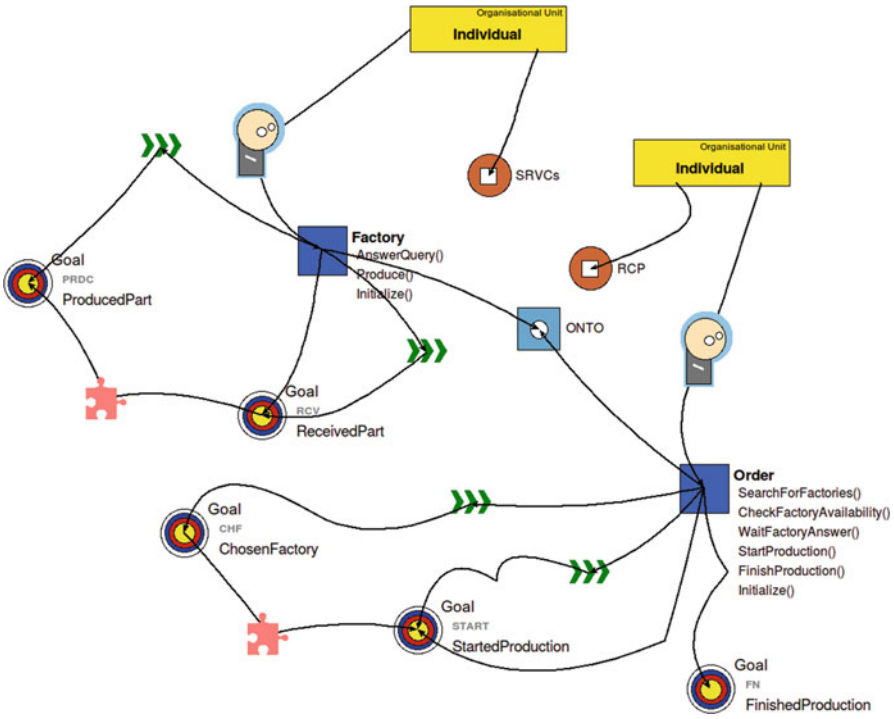


Fig. 10.4 Sample representation of the RecipeWorld by the organisational metamodel of this chapter

```
class AgentFactory(spade.Agent.Agent): [...]
  class AnswerQuery(): [...]
  class Produce(): [...]
  def initialise (): [...]
```

Further details from the model, e.g. knowledge, measurement and rewards of goals, are not shown in this example code, as they may be realised using various tools, most useful of which may be combination of Resource Description Framework (RDF) and Web Ontology Language (OWL), combined together into an ontology, since they are well adapted to the task of modelling knowledge.

10.3.2 A More Complex Example from an MMORPG

The second example that will be given in this chapter is devised to show how adaptable the proposed organisational metamodel is to the scale of the modelled system. The main difference between this example and the previous one is in scale,

e.g. in the number of possible roles within the system, in the number of active individual agents, in the number and diversity of tasks and goals and in the possible combinations of all the included elements of the system.

MMORPGs, and MMOGs as a more general concept, are a good application domain for LSMAS as they can engage hundreds, thousands and even millions of players. Many popular examples prove this (e.g. League of Legends, Hearthstone, Dota 2, World of Warcraft, etc.), and such games are still gaining popularity, presently having millions of regular active players [49]. Furthermore, MMOGs are interesting to research [33], since they are interesting to players that are eager to explore and interact with a virtual world, simultaneously motivating them to communicate and cooperate or fight with other players or elements of the environment. Group elements (including organisational features and social skills) are usually essential in games of MMORPG genre, since it is often impossible for a player's avatar (in-game character controlled by a human player) to survive or be successful in the given virtual world on their own.

The following example is based on an MMORPG scenario developed for the purposes of the Large-Scale Multi-Agent Modelling of Massively On-Line Role-Playing Games (ModelMMORPG) research project of the Artificial Intelligence Laboratory (AI Lab) of Faculty of Organisation and Informatics at the University of Zagreb. The scenario was based around a developed quest named The Quest for the Dragon Egg. In order to accomplish this quest, a player had to retrieve a Dragon Egg item from one of the three dragon dens located throughout the virtual Mana World, but the exact location of the egg was a secret, as was the precise time when the egg was going to spawn (with the interval being about 24 h). Upon finding the Dragon Egg item, after having fended off about a dozen Dragon monsters guarding it, a joint effort was necessary to transport the Dragon Egg item to the safe place. At least three player avatars (player characters) had to be present in each other's line of sight at any one moment while the Dragon Egg item was being transported, or the quest would fail. That was not the end though. In order to receive the main prize of the quest and to actually solve it, the egg had to be hatched using another special item, a hatching potion. This potion had to be made using several specific ingredients, making it another group effort. Only upon bringing the hatching potion and the Dragon Egg item, within a specified period of time, to a specific nonplayer character (NPC), a friendly Dragon monster could be spawned, and the quest finished. This devised quest is a clear representation of how important cooperation is in MMOGs, especially MMORPGs. It is important to note that only leaders of a group of players (called a party) could initiate the quest. Once the quest was initiated, only members of the initiating party could participate in the quest, and gain benefits of the solved quest. The key element in analysing the mentioned quest, and modelling it, is the fact that inclusion of many individual agents does not change the amount of set constraints in form of a role or any other concept.

An avatar is an individual agent in the scope of this example, so it shall be represented as an organisational unit. Since one of the key elements in the quest is a group of players, another organisational unit shall represent a party. Notice slightly different way of modelling, when compared to the previous example, since the

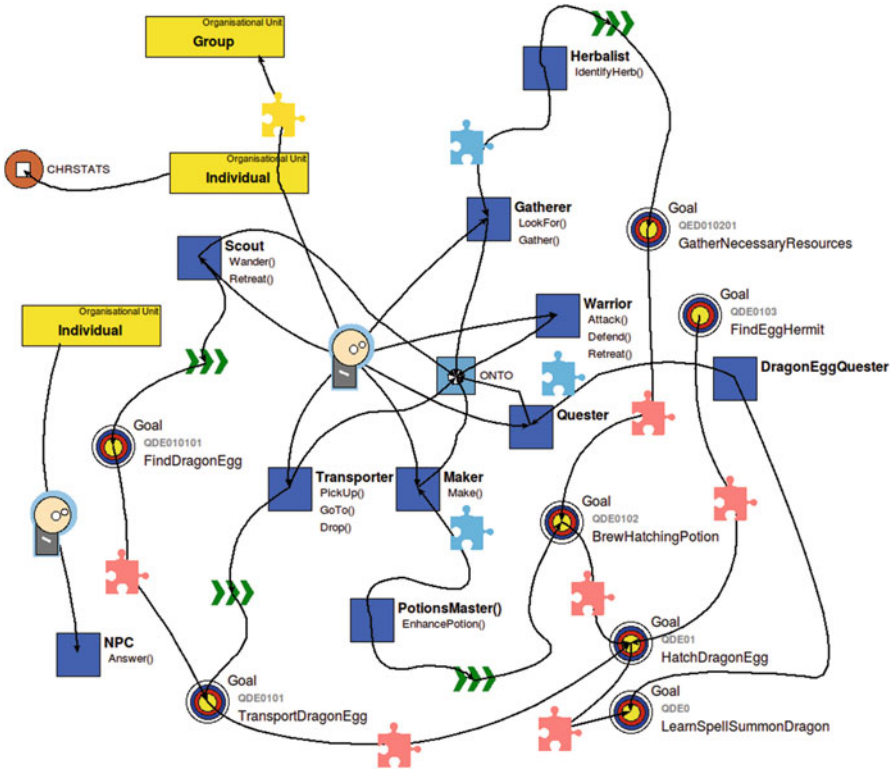


Fig. 10.5 Sample representation of the described quest for the Dragon Egg with modelled organisational units, roles, knowledge artefacts, goals and processes

emphasis will be on roles. Indeed, it is more interesting and useful to develop roles for the scenario described in this example. Furthermore, Fig. 10.5 shows only a part of the whole system, i.e. the part that describes elements, and their relationship, which are the closest to a player's avatar.

Individual agents are able to play several specific roles, e.g. gatherer, fighter, support, herbalist, transporter, scout, DragonEggQuester, etc.

Every role has specific processes it can use in order to achieve specific goals. The main objective of the quest is designed as an objective of the DragonEggQuester role and has a specified reward obtainable if finished. As such, the objective is divided into several lower-level specific quests. An individual agent is expected to use a role, and processes found therein, that is most suitable for reaching the identified specific goal.

State of the world, and of individual agents, is defined using knowledge artefacts. Specifics about various concepts found within the game are detailed in a connected ontology.

In addition to visual elements, some concepts have attributes that can receive specific values. Goals have such attributes denoting means of measurement if a goal

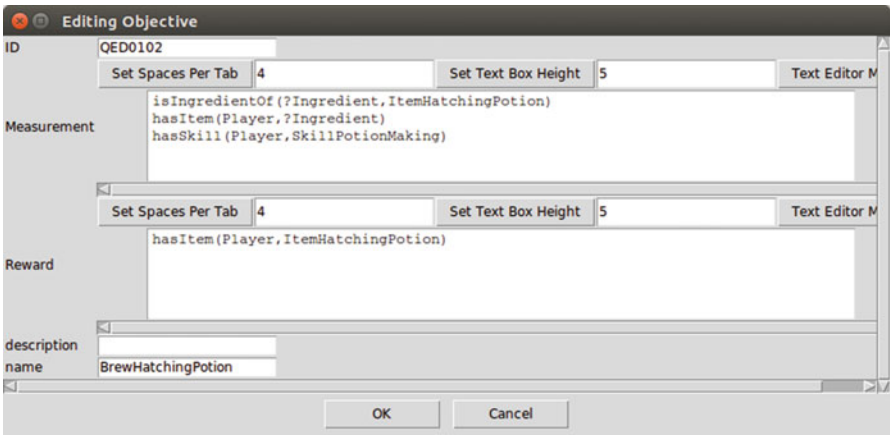


Fig. 10.6 Goal attributes and their values, where measurement and reward are motivated by the earlier mentioned seven perspectives of LSMAS organisational modelling

is satisfied and rewards for completing the given goal. The goal named BrewHatchingPotion shown in Fig. 10.5 has partially defined means of measuring if this particular goal is completed. The measurement is only partially defined in Fig. 10.6, since it does specify the necessary ingredients, but not the amount of these ingredients. Notice that both measurement and reward are given using a predicate logic expression. Therefore, reward predicate will be added to the main ontology upon finishing the goal, and measurement is based on already existing data. Depending on design decisions of the system developers, measurement can be based on data available from the world, an individual agent or a combination of both, expressed in the desired way.

10.4 Discussion

This section provides a brief discussion on the modelled examples, covering arguments backing up the presented metamodel, along with guidelines suggesting further development of the metamodel.

The examples described above show what the modelling process' result may look like, and the amount of information it may include. Laying out plans of the resulting system is certainly a worthwhile step when one is planning a new system and development thereof.

Working with distributed systems of large scale can create a state of potential unreliability, if the system being developed is not documented well enough. The work-in-progress metamodel that was shown in specific examples here is aiming to reduce the probability of such situations. Designed especially for modelling LSMAS, comprising concepts that may be used for that specific purpose, the metamodel offers system designers an easy and efficient way to lay down plans

of their systems-to-be and analyse the assembled-to-be model. Furthermore, it is possible to observe specific features that may inhibit usefulness, or success of the system being developed, and act accordingly.

From another perspective, metamodel showcased here can be used to facilitate easier or faster creation of a MAS specialised in distributed software project managements. The idea of an intelligent decision support and assisting system to be used by software project managers was laid out by Connor and Jenkins in [12].

Whichever of the above cases may be the prevalent one, managing distributed systems' development may prove to be easier and more efficient using the proposed metamodel, since not only does it provide an efficient overview of the modelled system, but it also makes generating basic computer code based on the defined model possible.

10.4.1 Evaluating the Proposed Approach to Modelling LSMAS

The proposed metamodel builds on recent studies of LSMAS organisational modelling and includes several concepts that are not found in the models mentioned in Sect. 10.2 of this chapter. Novel as it is, since it is based on an ontology comprising concepts of organisational modelling of LSMAS and it follows the recent trends in LSMAS development, it does have room for improvement.

Each of the modelled examples gives an insight about the metamodel from a specific perspective. The first example (Sect. 10.3.1) is about a simple system that requires no complex structures; therefore, it shows how the metamodel can be applied to small-scale systems. Short analysis of this example yields the following conclusions.

The model is expressive enough to represent the described system in as much detail as is needed for clear description of the given system. Since the metamodel being developed allows for various levels of abstractness, the model of example one could have been even more simple.

The two modelled organisational unit concepts could have been merged into one, but the metamodel is not yet expressive enough to distinguish between logical AND and logical OR connections – it would be advisable to model the individual agent using only one organisational unit that has access to either of the Individual KnArts and can play only one of the two modelled roles at any given time.

It is clear from the built model that the system comprises agents playing two different roles, with no constraints on communication possibilities. Each of the individual agents have access to some individual knowledge, and every role knows the same set of organisational knowledge. The goal hierarchy is clear and easy to understand, though the process concept is lacking insofar as it is not known what actions of a role are included in the modelled specific process. Individual processes may as well be represented using another model using the metamodel being

developed. Such an approach would further emphasise applicability of this metamodel to various levels of abstraction.

The second example, on the other hand, shows what a rather more complex system looks like being modelled using the proposed metamodel. Although a rather complex situation (that is a part of a larger world), it is easily and clearly modelled. Short analysis of this example yields the following conclusions.

An individual agent, modelled using an organisational unit concept, is defined using a slightly different approach from the one in example one. Only two types of individual agents are planned – agent commanded by a player and agent playing the role of a nonplayer character (NPC). Player avatars, as individual agents controlled by a player, can be grouped into a party (a group of player avatars). Furthermore, every individual avatar has some basic stats and their own inventory, as is shown using individual knowledge artefacts.

Diversity of roles is obvious, and their inclusion structure is clearly defined. The property denoting one role as a part of another role can be understood as an inheritance property; thus, actions defined for roles on higher level (e.g. Gatherer, Maker, Warrior, etc.) are inherited by roles on lower levels (e.g. Herbalist, PotionsMaster, SupportWarrior, etc.). It is therefore concluded that an organisational unit that can access a higher level role can also access a lower level role.

The main goal of the questing role is decomposed on several subgoals. As is visible from the model, the main goal structure is not completely linear, i.e. subgoal QDE01 is decomposed on three different goals with possible subgoals (e.g. QDE0101, QDE0102, QDE0103).

The model further shows that some of the subgoals can only be finished by a specific role (e.g. QED0102 named BrewHatchingPotion can only be finished by role named PotionsMaster). Naturally, every role has further constraints on when it can be played by a certain individual agent, but those constraints are not described in this particular model, nor at this stage of the metamodel development.

Reusability of concepts, as a feature of the proposed metamodel, is shown using various concepts of the model, yet it is most visible in joining roles to organisational knowledge artefacts. Two organisational knowledge artefacts, one representing the domain ontology and the other ontology comprising communication concepts, are modelled only once and are used by many different properties. Furthermore, roles do not have to be modelled more than once, but can be used by several different organisational units and by various properties.

The metamodel, as work-in-progress, can be evaluated as follows. The model is lacking in constraints of playing roles – it would seem that any individual agent may play any role at any given time. Although such a presumption may be true in the modelled part of the bigger system, it may be necessary to add constraint possibilities, e.g. implementing logic AND and OR expressions. This type of connection that would denote partial or complete inclusion of the concepts of property range may be useful in properties denoting hierarchical inclusion of concepts.

The two examples show how applicable the metamodel can be in situations of different size containing various elements. Even though the metamodel can be used for modelling the simple example, it is expressive enough to model the more complex example as well. The obvious problem in modelling the examples of this chapter is cluttered view and chaotic placement of numerous elements of the model. This is largely due to the very early stage of development of the model, since one of the features offered by the finished model is capability of modelling various levels of abstraction in different layers of the model, thus removing the visible clutter.

Since the metamodel is a work-in-progress, some planned features have not yet been implemented; wherefore, it does not completely comply with the modern trends of LSMAS modelling, i.e. perspectives of LSMAS modelling, as noted earlier, in Sect. 10.2.2. A notable feature missing is modelling of interorganisational dynamics, i.e. concepts for modelling the mentioned aspect of organisation. An introduction towards modelling organisations as individual agents is shown in example two, where there exists an organisational unit representing a group of individual agents. Interaction between such organisational units that represent groups may be modelled in the same layer, or level of abstraction, as the one shown in example two or in another one, represented by a new model.

The model, in its current state, partially or completely satisfies the following perspectives of LSMAS modelling mentioned in Sect. 10.2.2: organisational structure (since it supports modelling of decision and information flows), organisational culture (knowledge is modelled on individual or organisational level, norms are implemented using role concepts, and language is supported as a knowledge artefact, while goals do provide rewards upon being successfully completed), strategy (it is possible to model goals or objectives and their subgoals, i.e. how they relate to other goals and how goal success is measured), processes (every role can have defined activities it can perform, and those activities can be combined into a process that has a specific goal) and individual agents (it is possible to model individual agents insofar as to designate they exist, what knowledge they possess and what roles an individual can play). As mentioned earlier, organisational dynamics and context and interorganisational aspects have not been defined yet, although the first elements of these two perspectives are visible.

10.5 Conclusions

This chapter is about modelling LSMAS that conform to various elements of organisational design. A relevant set of perspectives of organisational architecture that is proposed to be used for modelling modern LSMAS was presented in a study by Schatten [47]. The work-in-progress metamodel presented in this chapter is based on the said set of perspectives and aims to provide a relevant upgrade of some recent studies of LSMAS organisational modelling.

The model in this chapter is presented as a suitable tool for planning and modelling LSMAS, in their many application domains, ranging from MMOGs to smart cities, smart transport and smart infrastructure, to distributed systems in general. Since planning and modelling phases have a great impact on the rest of the life cycle of a system, it is argued to be interesting to use a tool that allows one to create a model of the system being built and generate basis of the said system upon the defined model, which is a feature the metamodel presented in this chapter is intended to provide.

Outlined by the overview given in Sect. 10.2.1, the proposed metamodel should be expressive enough, yet simple to use, to provide the user with concepts that can model a simple example, as well as a more complex one. One of the main features of the proposed metamodel is recursive definitions of organisational units, goals and roles, as represented formally by Schatten [47]. Such an approach was shown in the second example (Sect. 10.3.2) where an organisational unit represents both an individual agent and a group of agents. Further examples are used for modelling goals and roles in Sect. 10.3.2 as well.

The author argues that using the proposed metamodel (once finished) may be beneficial for planning and modelling phases of development of distributed systems, as it allows the system developers to model the system in enough detail to represent functionality of the system in a way that is not overly complex and that is easy to read and comprehend. Therefore, although currently in development, and lacking in features, the proposed metamodel is argued to be a useful addition to recent research on LSMAS organisational modelling.

Future work concerning the proposed metamodel is clearly designated by the features lacking when compared to the perspectives of LSMAS organisational architecture used as the starting idea behind this metamodel. Further research into the existing models is needed, in form of a more thorough overview or analysis of those most recent, so as to compare the metamodel being developed (once finished) to those analysed. Based on the ontology pertaining selected organisational concepts for organisation of LSMAS, concepts included in the metamodel at this stage have to be analysed as well, in order to determine if some of the included concepts are redundant, or simply not needed, and there are concepts that have to be added. Furthermore, additional specific scenarios from an LSMAS application domain will be identified and the developed metamodel tested on them, so as to identify weak elements of the metamodel, and needed improvements.

Acknowledgements This work has been supported in full by the Croatian Science Foundation under the project number 8537.

References

1. Abbas HA (2014) Exploiting the overlapping of higher order. Int J Agent Technol Syst 6:32–57. doi:10.4018/ijats.2014070102

2. Abbas HA (2015) Realizing the NOSHAPE MAS organizational model. Int J Agent Technol Syst 7:75–104. doi:10.4018/IJATS.2015040103
3. Abbas HA, Shaheen SI, Amin MH (2015) Organization of multi-agent systems: an overview. Int J Intell Inf Syst 4:46–57. doi:10.11648/j.ijiis.20150403.11
4. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54:2787–2805. doi:10.1016/j.comnet.2010.05.010
5. Bădică A, Bădică C, Ganzha M, Ivanović M, Paprzycki M (2016) Experiments with multiple BDI agents with dynamic learning capabilities. In: Bajo J, Escalona JM, Giroux S, Hoffa-Dąbrowska P, Julián V, Novais P, Sánchez-Pi N, Unland R, Azambuja-Silveira R (eds) Highlights Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Int. Work. PAAMS 2016, Sevilla, Spain, June 1–3, 2016. Proc. Springer International Publishing, Cham, pp 274–286
6. Barriuso AL, de La Prieta F, Murciego ÁL, Hernández D, Herrero JR (2016) An intelligent agent-based journalism platform. In: Bajo J, Escalona JM, Giroux S, Hoffa-Dąbrowska P, Julián V, Novais P, Sánchez-Pi N, Unland R, Azambuja-Silveira R (eds) Highlights Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Int. Work. PAAMS 2016, Sevilla, Spain, June 1–3, 2016. Proc. Springer International Publishing, Cham, pp 322–332
7. Bergenti F, Iotti E, Poggi A (2016) Core features of an agent-oriented domain-specific language for JADE agents. In: de la Prieta F, Escalona JM, Corchuelo R, Mathieu P, Vale Z, Campbell TA, Rossi S, Adam E, Jiménez-López DM, Navarro ME, Moreno NM (eds) Trends Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Springer International Publishing, Cham, pp 213–224
8. Boomsma JJ, Franks NR (2006) Social insects: from selfish genes to self organisation and beyond. Trends Ecol Evol 21:303–308. doi:10.1016/j.tree.2006.04.001
9. Boulaire F, Utting M, Drogemuller R (2015) Dynamic agent composition for large-scale agent-based models. Complex Adapt Syst Model 3:1–23. doi:10.1186/s40294-015-0007-2
10. Van Den Broek EL, Jonker CM, Sharpanskykh A, Treur J, others (2006) Formal modeling and analysis of organizations. In: Boissier O, Padget J, Dignum V, Lindemann G, Matson E, Ossowski S, Sichman JS, Vázquez-Salceda J (eds) Coord. Organ. Institutions, Norms Multi-Agent Syst. Springer Berlin Heidelberg, pp 18–34
11. Bui N, Zorzi M (2011) Health care applications: a solution based on the internet of things. In: Proc. 4th Int. Symp. Appl. Sci. Biomed. Commun. Technol. – ISABEL'11. ACM Press, New York, pp 1–5
12. Connor RO, Jenkins J Using agents for distributed software project management. Management
13. Corkill DD, Lander SE (1998) Diversity in agent organizations. Object Mag 8:41–47
14. Coutinho LR, Sichman JS, Boissier O (2009) Modelling dimensions for agent organizations. In: Dignum V (ed) Handb. Res. Multi-Agent Syst. IGI Global, pp 18–50
15. Čyras K (2016) Argumentation-based reasoning with preferences. Bajo J, Escalona JM, Giroux S, Hoffa-Dąbrowska P, Julián V, Novais P, Sánchez-Pi N, Unland R, Azambuja-Silveira R Highlights Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Int. Work. PAAMS 2016, Sevilla, Spain, June 1–3, 2016. Proc. Springer International Publishing, Cham, 199–210
16. Decker KS (1996) TÆMS: a framework for environment centered analysis & design of coordination mechanisms. In: Found. Distrib. Artif. Intell. Wiley, pp 429–448
17. Dignum V (2004) A model for organizational interaction: based on agents, founded in logic. Utrecht University
18. Van Dyke Parunak H, Brueckner S (2001) Entropy and self-organization in multi-agent systems. In: Proc. Int. Conf. Auton. Agents. Montreal, Canada, pp 124–130
19. Van Dyke Parunak H, Odell J (2001) Representing social structures in UML. In: Proc. fifth Int. Conf. Auton. agents – AGENTS'01. ACM Press, New York, pp 100–101
20. Esteva M, Padget J, Sierra C (2002) In: Meyer J-JC, Tambe M (eds) Formalizing a language for institutions and norms. Springer, Berlin, pp 348–366

21. Fabretti A, Gärling T, Herzel S, Holmen M (2016) An agent-based model to study the impact of convex incentives on financial markets. In: Trends Pract. Appl. Scalable Multi-Agent Syst. PAAMS Collect. pp 3–13
22. Ferber J, Gutknecht O, Michel F (2004) From agents to organisations: an organizational view of multi-agent systems. Agent-Oriented Softw Eng IV:214–230. doi:10.1007/978-3-540-24620-6_15
23. Fontana M, Terna P (2015) From agent-based models to network analysis (and return): the policy-making perspective. Work Pap Ser 7:
24. Garcia-Rodriguez S, Sleiman HA, Nguyen V-Q-A (2016) A multi-agent system architecture for microgrid management. In: de la Prieta F, Escalona JM, Corchuelo R, Mathieu P, Vale Z, Campbell TA, Rossi S, Adam E, Jiménez-López DM, Navarro ME, Moreno NM (eds) Trends Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Springer International Publishing, Cham, pp 55–67
25. Gasser L (2001) Perspectives on organizations in multi-agent systems. In: Luck M, Mařík V, Štěpánková O, Trappl R (eds) Multi-agent Syst. Appl. Springer, Berlin, pp 1–16
26. Gliwa B, Koźlak J, Zygmunt A, Demazeau Y (2016) Combining agent-based and social network analysis approaches to recognition of role influence in social media. Demazeau Y, Ito T, Bajo J, Escalona JM Adv. Pract. Appl. Scalable Multi-agent Syst. PAAMS Collect. 14th Int. Conf. PAAMS 2016, Sevilla, Spain, June 1–3, 2016, Proc. Springer International Publishing, Cham, 109–120
27. Hadfi R, Ito T (2016) Holonic multiagent simulation of complex adaptive systems. In: Bajo J, Escalona JM, Giroux S, Hoffa-Dąbrowska P, Julián V, Novais P, Sánchez-Pi N, Unland R, Azambuja-Silveira R Highlights Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Int. Work. PAAMS 2016, Sevilla, Spain, June 1–3, 2016. Proc. Springer International Publishing, Cham, 137–147
28. Hernández D, Villarrubia G, Barriuso AL, Lozano Á, Revuelta J, De Paz JF (2016) Multi agent application for chronic patients: monitoring and detection of remote anomalous situations. Bajo J, Escalona JM, Giroux S, Hoffa-Dąbrowska P, Julián V, Novais P, Sánchez-Pi N, Unland R, Azambuja-Silveira R Highlights Pract. Appl. Scalable Multi-Agent Syst. PAAMS Collect. Int. Work. PAAMS 2016, Sevilla, Spain, June 1–3, 2016. Proc. Springer International Publishing, Cham, 27–36
29. Horling B, Lesser V (2005) A survey of multi-agent organizational paradigms. Knowl Eng Rev 19:281. doi:10.1017/S0269888905000317
30. Hübner JF, Sichman JS, Boissier O (2002) A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: Bittencourt G, Ramalho GL (eds) Adv. Artif. Intell. Springer, Berlin, pp 118–128
31. Hübner JF, Vercouter L, Boissier O (2009) Instrumenting multi-agent organisations with artifacts to support reputation processes. In: Hübner JF, Matson E, Boissier O, Dignum V (eds) Coord. Organ. Institutions Norms Agent Syst. IV. Springer, Berlin, pp 96–110
32. Kir H, Erdoğan N (2016) Agent-based semantic business process management methodology. Demazeau Y, Ito T, Bajo J, Escalona JM Adv. Pract. Appl. Scalable Multi-agent Syst. PAAMS Collect. 14th Int. Conf. PAAMS 2016, Sevilla, Spain, June 1–3, 2016, Proc. Springer International Publishing, Cham, 145–156
33. Lofgren ET, Fefferman NH (2007) The untapped potential of virtual game worlds to shed light on real world epidemics. Lancet Infect Dis 7:625–629. doi:10.1016/S1473-3099(07)70212-8
34. Losilla J, Olivares T, Fernández-Caballero A (2016) Multi-agent-based framework for prevention of violence against women: scenarios in Google Maps. In: de la Prieta F, Escalona JM, Corchuelo R, Mathieu P, Vale Z, Campbell TA, Rossi S, Adam E, Jiménez-López DM, Navarro ME, Moreno NM (eds) Trends Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Springer International Publishing, Cham, pp 277–285
35. McCauley L, Franklin S (2002) A large-scale multi-agent system for navy personnel distribution. Connect Sci 14:371–385. doi:10.1080/0954009021000068934

36. Mihaylov M, Razo-Zapata I, Rădulescu R, Jurado S, Avellana N, Nowé A (2016) Smart grid demonstration platform for renewable energy exchange. Demazeau Y, Ito T, Bajo J, Escalona JM Adv. Pract. Appl. Scalable Multi-agent Syst. PAAMS Collect. 14th Int. Conf. PAAMS 2016, Sevilla, Spain, June 1–3, 2016, Proc. Springer International Publishing, Cham, 277–280
37. Okreša Đurić B (2016) Organizational metamodel for large-scale multi-agent systems. de la Prieta F, Escalona MJ, Corchuelo R, Mathieu P, Vale Z, Campbell AT, Rossi S, Adam E, Jiménez-López MD, Navarro EM, Moreno MN Adv. Intell. Syst. Comput Springer International Publishing, Seville, 387–390
38. Okreša Đurić B, Konecki M (2015) Modeling MMORPG players' behaviour. In: Hunjak T, Kirinić V, Konecki M (eds) Cent. Eur. Conf. Inf. Intell. Syst. 2015. Faculty of Organization and Informatics, Varaždin, HR, pp 177–184
39. Okreša Đurić B, Schatten M (2016) Defining ontology combining concepts of massive multiplayer online role playing games and organization of large-scale multi-agent systems. 39th Int. Conv. Inf. Commun. Technol. Electron. Microelectron.
40. Parrish JK, Viscido SV, Grünbaum D (2002) Self-organized fish schools: an examination of emergent properties. Biol Bull 202:296–305. doi:10.1016/j.foodchem.2006.01.008
41. Picard G, Hübner JF, Boissier O, Gleizes M-P (2009) Reorganisation and self-organisation in multi-agent systems. In: Int. Work. Organ. Model. Paris, pp 66–80
42. Pico-Valencia P, Holgado-Terriza JA (2016) ADELE: a middleware for supporting the evolution of multi-agents systems based on a metaprogramming approach. In: de la Prieta F, Escalona JM, Corchuelo R, Mathieu P, Vale Z, Campbell TA, Rossi S, Adam E, Jiménez-López DM, Navarro ME, Moreno NM (eds) Trends Pract. Appl. Scalable Multi-Agent Syst. PAAMS Collect. Springer International Publishing, Cham, pp 297–310
43. Posey RB, Haire M (1961) Modern organization theory. Adm Sci Q 5:609–611. doi:10.2307/2390625
44. Román JA, Rodríguez S, de la Prieta F (2016) Improving the distribution of services in MAS. Bajo J, Escalona JM, Giroux S, Hoffa-Dąbrowska P, Julián V, Novais P, Sánchez-Pi N, Unland R, Azambuja-Silveira R (eds) Highlights Pract. Appl. scalable multi-agent Syst. PAAMS Collect. Int. Work. PAAMS 2016, Sevilla, Spain, June 1–3, 2016. Proc. Springer International Publishing, Cham, 37–46
45. Russell SJ, Norvig P (2010) Artificial intelligence: a modern approach, 3rd edn. Prentice Hall, Englewood Cliffs
46. Schatten M (2012) Complex analytical method for self-organizing multiagent systems. In: Cent. Eur. Conf. Inf. Intell. Syst. 2012. Faculty of Organization and Informatics, Varaždin, pp 63–70
47. Schatten M (2014) Organizational architectures for large-scale multi-agent systems' development: an initial ontology. Adv Intell Syst Comput 290:261–268. doi:10.1007/978-3-319-07593-8_31
48. Schatten M, Ševa J, Tomičić I (2016) A roadmap for scalable agent organizations in the Internet of Everything. J Syst Softw 115:31–41. doi:10.1016/j.jss.2016.01.022
49. Schatten M, Tomicic I, Okreša Đurić B (2015) Multi-agent modeling methods for massivley multi-player on-line role-playing games. In: Biljanović P (ed) 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. IEEE, Opatija, HR, pp 1256–1261
50. Scheutz M (2010) A multi-agent system infrastructure for large-scale autonomous distributed real-time intelligence gathering systems. Proc. ISCA
51. Tomičić I (2016) Agent-based framework for modelling and simulation of resource management in smart self-sustainable human settlements. University of Zagreb
52. Tomičić I, Schatten M (2015) Towards an agent based framework for modelling smart self-sustainable systems. Interdiscip Descr Complex Syst 13:50–63. doi:10.7906/indecs.13.1.7
53. Tomičić I, Schatten M (2016) Agent-based framework for modeling and simulation of resources in self-sustainable human settlements: a case study on water management in an eco-village community in Croatia. Int J Sustain Dev World Ecol 1–10. doi: 10.1080/13504509.2016.1153527

54. Tsarev A, Skobelev P (2016) Multi-agent supply scheduling system prototype for energy production and distribution. Demazeau Y, Ito T, Bajo J, Escalona JM Adv. Pract. Appl. scalable multi-agent Syst. PAAMS Collect. 14th Int. Conf. PAAMS 2016, Sevilla, Spain, June 1–3, 2016, Proc. Springer International Publishing, Cham, 290–293
55. Vermesan O, Friess P, Guillemin P, Gusmeroli S, Sundmaeker H, Bassi A, Jubert IS, Mazura M, Harrison M, Eisenhauer M, Doody P, Peter F, Patrick G, Sergio G, Harald, Sundmaeker Alessandro B, Ignacio Soler J, Margaretha M, Mark H, Markus E, Pat D (2009) Internet of things strategic research roadmap. In: Vermesan O, Friess P, Guillemin P, Gusmeroli S, Sundmaeker H, Bassi A, Jubert IS (eds) Internet Things Strateg. Res. Roadmap. pp 9–52
56. Villavicencio C, Schiaffino S, Diaz-Pace JA, Monteserin A, Demazeau Y, Adam C (2016) A MAS approach for group recommendation based on negotiation techniques. In: Demazeau Y, Ito T, Bajo J, Escalona JM (eds) Adv. Pract. Appl. Scalable Multi-agent Syst. PAAMS Collect. 14th Int. Conf. PAAMS 2016, Sevilla, Spain, June 1–3, 2016, Proc. Springer International Publishing, Cham, pp 219–231
57. Vlacheas P, Giaffreda R, Stavroulaki V, Kelaidonis D, Foteinos V, Poulios G, Demestichas P, Somov A, Biswas A, Moessner K (2013) Enabling smart cities through a cognitive management framework for the internet of things. IEEE Commun Mag 51:102–111. doi:10.1109/MCOM.2013.6525602
58. Weyns D, Haesevoets R, Helleboogh A (2010) The MACODO organization model for context-driven dynamic agent organizations. ACM Trans Auton Adapt Syst 5:1–29. doi:10.1145/1867713.1867717
59. Weyns D, Haesevoets R, Helleboogh A, Holvoet T, Joosen W (2010) The MACODO middleware for context-driven dynamic agent organizations. ACM Trans Auton Adapt Syst 5:1–28. doi:10.1145/1671948.1671951
60. Wikarek J, Sitek P (2016) A multi-level and multi-agent approach to modeling and solving supply chain problems. In: Bajo J, Escalona JM, Giroux S, Hoffa-Dąbrowska P, Julián V, Novais P, Sánchez-Pi N, Unland R, Azambuja-Silveira R (eds) Highlights Pract. Appl. Scalable Multi-Agent Syst. PAAMS Collect. Int. Work. PAAMS 2016, Sevilla, Spain, June 1–3, 2016. Proc. Springer International Publishing, Cham, pp 49–60
61. Yu H, Shen Z, Leung C (2013) From internet of things to internet of agents. In: 2013 I.E. Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Soc. Comput. IEEE, pp 1054–1057
62. Žugaj M, Schatten M (2005) Arhitektura suvremenih organizacija. Tonimir i Fakultet organizacije i informatike, Varaždinske
63. Žugaj M, Šehanović J, Cingula M (2004) Organizacija, 2nd edn. TIVA Tiskara Varaždin, Varaždin

Part III
Advances in Software Project Management and Distributed Software Development

Chapter 11
Optimizing Software Error Proneness Prediction Using Bird Mating Algorithm

Amrit Pal, Harsh Jain, and Manish Kumar

11.1 Introduction

Software engineering is the process of designing, developing, and maintaining a software product. It deals with complete life cycle of software production, beginning with requirement collection to the last phase of maintenance [1]. Apart from developing the software, it is important to estimate its size and cost. Also it is necessary to take an account of errors and defects that might influence the functionality. Errors and defects can be referred to a piece of code in software which adversely affects functional and nonfunctional requirements. Testing is the phase which is responsible for finding out errors and defects and removing them.

With the advancement in computer science and software engineering, the complexity of software is increasing day by day. Due to the increased complexity, a number of errors are also increasing in development process [2]. Having planned a software in an organized way also does not allow the engineers to escape the errors. However, occurrence of errors can also be considered as necessary for software development, because they provide a check for quality of the software.

Prediction of software proneness to contain errors is one of the crucial issues that need to be taken care of. It helps a lot to save cost and time in the development of software. However, finding an appropriate method and model for predicting software error proneness has been the main goal of engineers nowadays. One of the useful models is machine learning that can be used for predicting error proneness. There are several methods available in machine learning like artificial neural network, support vector machine, k-means clustering, etc. that can be employed for prediction purpose. For the purpose of predicting software error proneness in

A. Pal • H. Jain (✉) • M. Kumar
Department of Information Technology, Indian Institute of Information Technology, Allahabad, India
e-mail: jharsh38@gmail.com

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*, Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_11

this work, artificial neural network is used. It is a multilayer feed-forward network which contains neurons and connections, and each connection is associated with a weight. These weights are adjusted in the learning process of neural network. When artificial neural network is trained using conventional algorithm like backpropagation algorithm, it suffers from some serious problems, e.g., local minima and overfitting of data. To solve these problems, optimization algorithms may be used, e.g., genetic algorithm, bird mating optimizer algorithm, etc. In the process of predicting software error proneness, two evolutionary algorithms genetic algorithm and bird mating optimizer algorithm are used to train neural network.

Promise repository provides five datasets – cm1, jm1, kc1, kc2, and pc1 [3]. These datasets contain data of software which is attributed by different software metrics and predict whether the software is error prone. It is a two-class classification problem, where classes are true or false. Artificial neural network is employed to classify the data into two classes. Before training of neural network, genetic algorithm is used to select the features of importance, i.e., those variables which are more valuable in comparison to others. Feature extraction helps reduce the complexity of the network and better results. Once the features have been selected, the hybrid algorithms are implemented. First genetic algorithm is employed to optimize the weights of artificial multilayer feed-forward neural network. Another algorithm is bird mating optimizer which is also used to optimize the weights of neural network. The results of above two methods are analyzed to reach to a particular model of prediction.

11.1.1 Error, Fault, and Failure

In software testing, inconsistency within software can be termed as error, fault, bug, failure, or defect. These terms apparently sound synonymous but they are not [4]. They are dissimilar and have entirely different meanings. The relationship between these terms is clearly depicted in Fig. 11.1 and as described below:

- *Error:* can be best expressed as "to err is human." Errors are mistakes that are made by humans in source code of the modules of a software.
- *Fault:* is basically a reward for an error. It comes when an error is made in the program. Fault represents an incorrect step or flow of the program.
- *Bug:* is a proof that represents a fault has occurred in the program. It represents the unintended behavior of the module.
- *Failure:* occurs when the fault executes. It represents the inability of a module or program that it cannot perform its desired function.
- *Defect:* represents a software cannot fulfill its functional requirements. It is the result of a failure which has occurred in response to an error.

The following example shows how a mistake in source code can lead to incorrect results. A module in C language is written, which is a part of software that calculates and returns the product of two numbers which are passed to it.

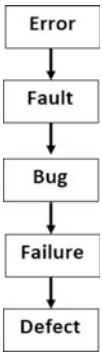


Fig. 11.1 Error, fault, and defect

```
int multiply (int a, int b)
{
   int prod;
   prod = a/b;
   return prod;
}
```

Main() function calls this module using the function call *multiply(6,3)*. The module *multiply(int, int)* returns two as output, which is not the desired answer. The correct output should be 18 according to the product rule. This module gives incorrect result in response to an error where division is performed instead of multiplication. This instance represents a fault in the program, and a failure occurs when this module executes. The software is said to be defective because it does not perform its intended function.

The above given example is the simplest module that can ever have an error. But these days software products are more complex and so are their modules. They might contain a lot of errors and faults. Identifying and removing errors from a module is a very rigorous and exhaustive task, but prediction about the possibility of a module to contain errors is even more challenging. Because once the error and its type are known, it is comparatively easy to debug a module than make prediction about it when there is no information about the error. Prediction about the error proneness is made using the past data of some of the important software metrics which help decide whether the modules might contain errors or not. Properties of a module which help in the prediction process include:

1. Lines of code
2. Halstead metrics
3. Cyclomatic complexity
4. Design complexity
5. Essential complexity

Thus, it can be very helpful in the development and maintenance of software if the error proneness of a module can be detected at early stages of the development process.

This chapter comprises of five more sections. Section 11.2 describes the past work which is related to software error proneness prediction, artificial neural network, and optimization algorithms. Section 11.3 comprises of the prediction model and its three components: variable subset selection, GANN, and BMANN. Section 11.4 describes the dataset used for experimentation purpose and the complete setup of experiment. Section 11.5 shows the analysis of the performance of prediction model, and the final section focuses on conclusion and future work.

11.2 Related Work

11.2.1 Artificial Neural Network

There have been several researches carried out which are related to training of neural network. To train multilayer feed-forward network, an error backpropagation method can be used [5]. A technique was proposed to increase rate of convergence of artificial neural network in [6]. It highlighted the facts that method of steepest descent is slow and various reasons behind it. To overcome the problem, the paper proposed four heuristics suggesting that every weight of network should be given its own learning rate and that these rates should be allowed to vary over time.

Artificial neural network is a computation model which is inspired by the nervous system and its working inside human body [7]. Neural network is a set of connected input/output units where each connection between the units has a weight associated to it as shown in Fig. 11.2. Backpropagation algorithm is applied over multilayer feed-forward network. Multilayer feed-forward network consists of three layers, namely, input layer, hidden layer, and output layer. It consists of one input layer, one or more hidden layers, and an output layer. Each layer has several neurons/processing units, which processes the data and produces the output.

The processing unit in each layer accepts inputs from the neurons of the previous layer; inputs to the neurons of input layer are fed by the user and process them. The processing unit applies activation function over the weighted input and produces output [8] (Fig. 11.3).

Here $x_1, x_2, \ldots, x_n$ are input values and $w_1, w_2, \ldots, w_n$ are corresponding weight values. These inputs and weights are multiplied correspondingly, and their sum is found using Eq. 11.1.

$$\text{net}_j = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n \tag{11.1}$$

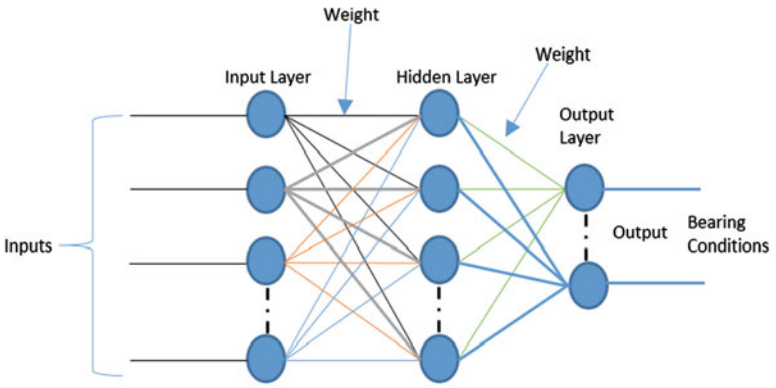


Fig. 11.2 Artificial neural network

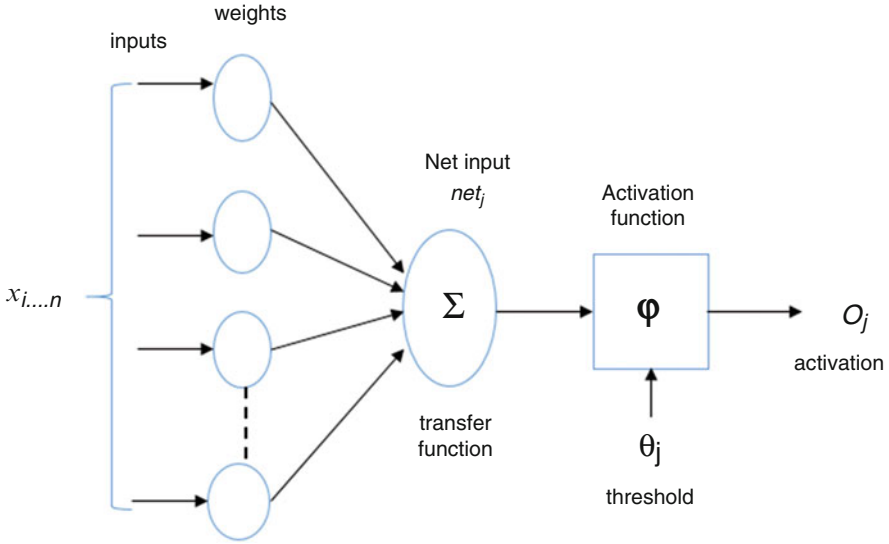


Fig. 11.3 Perceptron model (Adopted from Ref. [8])

θ_j represents threshold which is applied to the processing unit. This unit applies activation function over net input net_j and threshold θ_j to produce output o_j. Some of the activation functions are:

- Step function
- Linear function
- Log-sigmoid function
- Tan-sigmoid function

Neural networks are highly advantageous to solving such problems which include noisy data because they can easily predict the class for the instance which they are not trained for. They are applicable on real world data which is related to different fields, e.g., industrial data, hospital data, historical data and financial transactions, etc.

One of the most popular methods to train the neural network is backpropagation which was introduced in [5]. Backpropagation algorithm works in two phases: learning phase and classification phase. During learning phase, training dataset is fed into the network through input layer neurons. Input from input layer is directly passed as the output of input layer and is presented as the input to first hidden layer. All the hidden layers process their inputs using activation function and the weights associated to them. Output of last hidden layer becomes input for the output layer. Output layer neurons process the input provided to them and produces the output. Calculated output is compared against the actual output and error is calculated. In learning process this error is minimized by updating the weights with the consideration of a factor called learning rate. Finally when the network is trained, it goes through the testing phase and performs classification over testing data. In the training dataset, the classes of data are already known, so this algorithm is a supervised learning algorithm. Supervised learning consists of dataset where class labels are known prior to the classification.

Neural network can be employed to solve many software engineering problems. Neural network can be applied to various fields, e.g., autonomous vehicle control, software effort estimation, software defect prediction, etc. Neural network is seen as a highly nonlinear function. So training problem can be thought of as a general function optimization problem, where weights and biases are to be adjusted. In that case Levenberg-Marquardt algorithm [15] can be applied. It is a simple and robust algorithm which is used for approximating a function. It is used to approach second-order training speed without computing Hessian matrix. There are various issues that neural network suffers from, e.g., local minima problem and overfitting of data.

An approach that improves backpropagation algorithm with the help of adaptive gain is shown in [9]. The paper states that most of the gradient-based algorithms use negative gradient of error for optimization of weights as a direction. This paper presents the method to improve the efficiency of training algorithm by adaptively modifying the search direction. The activation function of this algorithm uses the value of gain parameter which adaptively modifies the search direction. Comparative results have been shown on popular classification problems, Wisconsin breast cancer classification problem and Iris classification problem. Results show that the training efficiency of backpropagation algorithm gets increased by using proposed method.

There are various other ways to train the neural network such as genetic algorithm [10]. The paper proposed a method to combine the two approaches, genetic algorithm and artificial neural network. It states that genetic algorithm might help determine the structure of neural network, i.e., hidden layers and output layer and weights of the neural network. It deals with finding the weights only and does not go into the detail of restructuring the neural network. The paper presented

a method where genetic algorithm is employed to determine the weights of the network. The results of this approach are compared with the results that are obtained when the neural network is trained by backpropagation algorithm. The comparison shows that neural network along with genetic algorithm produces better results.

11.2.2 Software Error Proneness Prediction

A method for software defect prediction has been proposed in [11] which used classification tree on two datasets of NASA. Classification trees are very effective to identify risky components. Basically, classification trees depend on metrics to classify the data. For example, metrics which can be used to classify the data are line of code, cyclomatic complexity, number of operators and operands, volume, length, etc. It is an iterative process where the process is repeated at every node to make decisions. This method proves to be efficient in the area of software defect prediction, and results show the precision of 79.3%.

A high-performance neural network for software defect prediction was proposed in [12]. It used a hybrid concept of support vector machine and artificial neural network to predict the software defects on datasets provided by Promise repository. The proposed methodology overcomes the issue of overfitting and also maximizes the margin of classification. Proposed model is applied on three datasets which are provided by NASA. Results of the proposed model have been compared with the results of several other machine learning algorithms, e.g., KNN, RBF, C4.5, NB2, and RIPPER.

Another technology was presented in [13] in which radial basis function has been used to train neural network for software error proneness prediction. Results are interpreted by using receiver operating characteristics analysis which uses the values of area under the curve, sensitivity, and a cutoff threshold. The proposed method is applied on the project and issue tracking system which is provided by verification and validation program of NASA. Defects are classified into five levels of severity from level 1 as most severe defect to level 5 as least severe defect. The results show that the method works more efficient for level 1 defects than for level 5 defects.

Software defect prediction model with the help of fuzzy system was presented in [14]. It introduces a bell-shaped function in hidden layer of the network which is based on fuzzy logic. The neural network is trained with two hidden layers; first hidden layer employs tan-sigmoid function, whereas second hidden layer employs bell-shaped function. Fuzzy logic is advantageous in the sense that it works on vagueness as a human brain works. So it can deal with vague concepts in order to make decisions. It presents comparative results of error proneness prediction on datasets provided by NASA in Promise repository. It compares the results of proposed method with the results of other algorithms like backpropagation algorithm, logistic regression and random tree, etc.

Another famous algorithm to train the neural network known as Levenberg-Marquardt (LM) algorithm was used to predict the software error proneness in [15]. The study of this paper suggests that neural network based on LM algorithm provides better accuracy which is up to 88.09% as compared to other neural network algorithm such as neural network based on linear function, neural network based on polynomial function, and neural network based on quadratic function.

11.2.3 Optimization Algorithms

In the area of predicting software error proneness using neural network, many research papers have been presented that focused on optimization of neural network using various algorithms such as genetic algorithm, artificial bee colony algorithm, particle swarm optimization, etc. A technique to optimize the process is proposed in [16], which uses a hybrid approach of genetic algorithm and bagging technique to predict the software defects. The method uses genetic algorithm for optimizing the parameters of network and bagging technique to solve the problem of class imbalance. They applied the proposed methodology on several datasets of NASA Promise repository and obtained efficient results. Results show that the method has obtained a higher accuracy, so it concludes that the proposed methodology improves the performance of neural network.

A cost-sensitive strategy to predict software error proneness is proposed in [17] which suggests consideration of cost issues while developing the model. Earlier methods do not consider the issue of cost which occurs in case of misclassifying the defect-prone software modules. This paper considers three cost-sensitive boosting algorithms which were used to boost the neural network. The first algorithm is based on threshold, where threshold for classification is moved toward non-defect-prone software modules, in order to predict the defect-prone software modules correctly. The other two algorithms are based on weight-update rule. The data samples which are related to misclassified defect-prone software modules are given boosted weights in order to improve the cost of the procedure. The performances of the proposed algorithms are compared on the data of NASA repository, and normalized expected cost of misclassification is calculated. The results show that the method of threshold works more effectively than other two methods.

Another cost-sensitive technique is presented in [18] which employs a hybrid approach of artificial bee colony algorithm and neural network. In this approach neural network has been trained by artificial bee colony algorithm to obtain optimal weights. Artificial bee colony optimization intends to optimize the parameters of false-positive rate and false negative rate which are multiplied by coefficients of parametric cost. This approach is applied to the datasets of NASA which are available in Promise repository. The results of the proposed methodology are calculated in the following terms: normalized expected cost of misclassification, probability of false alarm, accuracy, probability of detection, and area under the curve.

There is a latest optimization algorithm "bird mating optimizer" which is proposed by Askarzadeh and Rezazadeh [19] to find the optimal weights of neural network. The bird mating algorithm is used to optimize the training of artificial neural network. They apply the model to some popular classification problems like iris flower, Pima Indian diabetes, and Wisconsin breast cancer. So based on the above survey, bird mating optimizer algorithm with neural network can be used to model the software defect prediction system.

11.3 Prediction Model

Software error proneness prediction model is built with the help of artificial neural network. Neural network is implemented using two hybrid methods: neural network using genetic algorithm and neural network using bird mating optimizer algorithm. Once both the methods are applied, their performances are analyzed. The overall proposed model can be broken down into a series of different tasks as shown in Fig. 11.4 and presented below:

1. Selection of useful variables with the help of genetic algorithm and mutual information.
2. GANN – Artificial neural network is trained by genetic algorithm. Predicting software error proneness using a hybrid approach of genetic algorithm and neural network learning.
3. BMANN – Artificial neural network is trained by bird mating algorithm. Predicting software error proneness using a hybrid approach of bird mating optimizer and neural network learning.

11.3.1 Variable Subset Selection

Variable subset selection can also be known as feature selection and attribute selection. It is a preprocessing task, where complexity of data is decreased by reducing the number of features or attributes [20]. The dataset, which is given for a particular task, contains a number of attributes, out of which certain attributes are relevant, i.e., either they are of no use or they provide redundant information. It is suggested that if those irrelevant attributes are removed before beginning the actual process, it might reduce the complexity of process and increase the performance of model.

Hanchuan Peng et al. proposed an algorithm which can be used for feature selection in [21]. The algorithm is based on the principle of minimum redundancy and maximum relevance between the attributes and target class. The relevance and redundancy can be computed with the help of mutual information, distance score,

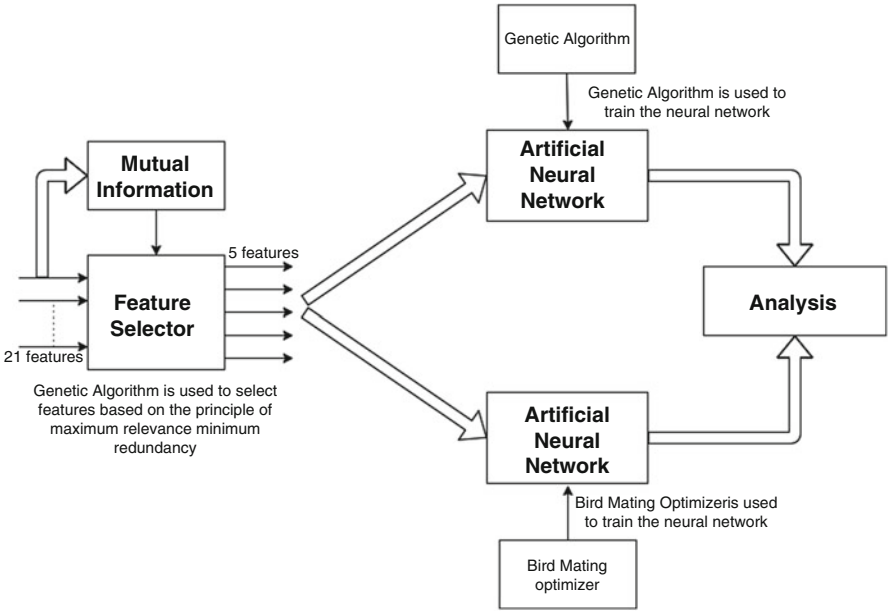


Fig. 11.4 Prediction model

correlation, or similarity scores. The overall objective of the method is to maximize the relevance between the features and target class, and reducing the redundancy between the features [20, 21].

The relevance between the set of feature and the target class can be calculated as follows:

$$D(S,c) = \frac{1}{|s|}\sum_{f_i \in s} I(f_i;c) \tag{11.2}$$

where S is the set of features, c represents the class vector, f_i represents the individual feature, and I stand for mutual information.

The redundancy between the features of a set can be calculated as follows:

$$R(S) = \frac{1}{|s|^2}\sum_{f_i, f_j \in s} I\left(f_i;f_j\right) \tag{11.3}$$

where f_i and f_j are the features of set S.

The mRMR criterion can be computed by the following formula:

$$\text{mRMR} = \max(D(S,c) - R(S)) \tag{11.4}$$

The mRMR criterion defines the suitability and optimality of the subset which is selected. This criterion can be optimized by selecting different subset of variables.

This task has been achieved with the help of genetic algorithm in the proposed model of this work.

The dataset provided by NASA contains 21 variables on which software error proneness depends. But it is not relevant to use all the 21-variable attributed dataset as it would lead to a complicated computation and produce improper results. There are certain attributes which are important and relevant to classify the software into respective categories, whereas remaining attributes do not contribute much to the model of error proneness prediction.

In this approach, mutual information is calculated in four different ways: Hx, mutual information of each feature; Hy, mutual information of output feature; MIxy, mutual information between one input feature and output feature; and MIxx, mutual information between each pair of input features. These four sets of mutual information are used to select those features that have maximum relevance with output feature. Mutual information between two random variables x and y is calculated using probability density functions of variables $p(x)$ and $p(y)$ and joint probability density function $p(x,y)$ using Eq. 11.5.

$$I(x;y) = \iint p(x,y)\log\frac{p(x,y)}{p(x)p(y)}dxdy \tag{11.5}$$

Genetic algorithm is used to optimize the solution. Initial population consists of a large number of chromosomes, i.e., 200 times the number of features to be selected. For example, if five variables are to be selected, then initial population consists of 1000 chromosomes. Each chromosome consists of random integer numbers within the range of indices of features. Hx, Hy, MIxy, and MIxx are calculated using the above given formula. The fitness function of genetic algorithm is based on the principle of maximum relevancy and minimum redundancy, i.e., a feature should be highly relevant to target class and there should be minimum redundancy between the features [21]. Fitness value of each chromosome is calculated using the given formula in Eq. 11.6.

$$S = \text{mean\_MIxy} - c*\text{mean\_MIxy} \tag{11.6}$$

where, mean_MIxy denotes mean relevance of the chromosome, i.e., mean value of mutual information, MIxy, between input feature and target class, and mean_MIxx denotes the mean redundancy of the chromosome, i.e., mean value of mutual information, MIxx, between input features. It is a maximization function as the function has to be maximized by genetic algorithm to find relevant features. The higher the value of fitness function, the better is the quality of chromosome. Each individual chromosome is ranked based on its fitness value. Now crossover is performed between two parent individuals. Two indices of chromosomes, pai and mae, are selected randomly using asymmetric distribution [22]. Crossover is performed using the pseudocode of module crossover_feature_selection().

```
crossover_feature_selection()
for i = 1 to req_features
   if r > 0
      offspring(i) = pai(i)
   else
      offspring(i) = mae(i)
```

where r is random number between 0 and 1. After performing crossover for every individual, the fitness value of each offspring is calculated. A parent chromosome is replaced by its offspring if its fitness value is less than the fitness value of offspring and a new population is generated. This process is iterated until a maximum number of generations are produced or an optimal solution is achieved.

11.3.2 GANN: Genetic Algorithm-Based Neural Network

Genetic algorithm is inspired by the process of natural evolution [23]. It is an adaptive heuristic search algorithm that is inspired by and based on the idea of natural selection and genetics. Genetic algorithm provides a general solution and is applicable to any search space. It is applied in the areas of which very little is known. The only information needed is that which of the solutions will be good to solve the problem. It uses the principle of evolution and search for an optimal solution out of several given solutions to a problem. Genetic algorithm is mainly used to solve optimization problem. It is used to simulate the process of natural evolution which is based on the principle of "survival of fittest."

In genetic algorithm, following terminology is used:

- Individual: It represent a possible solution to the given problem.
- Population: It is the collection of all individuals.
- Search space: It refers to the space which contains all the possible solutions.
- Chromosome: It represents a set of properties of each individual
- Selection: Selection of an individual according to a predefined criterion.
- Fitness function: It is a function which describes fitness of each individual.
- Crossover: A process where two individual exchange their chromosomes.
- Mutation: A process to randomly modify individual.

Genetic algorithm works upon a population of individual where each individual represents a possible solution to the given problem [24]. Each individual has a set of properties which are referred to as chromosomes. These individuals are encoded using several encodings, e.g., binary encoding, real-number encoding, etc. Genetic algorithm heads toward an optimal solution in the search space where the optimality of each solution is defined by the fitness function. Fitness function is a criterion that defines the extent to which the solution is fit for the problem.

It is an iterative process as shown in Fig. 11.5, which is started by initializing a random population of individuals and in each iteration a population is called a generation. In each iteration, fitness function is applied over each individual to calculate the fitness. Once the fitness of every individual is obtained, some individuals are selected that best suit the solution of the problem. Individuals can be selected via different mechanisms such as rank selection method, roulette wheel selection, etc. After selecting the individuals, two genetic operators, crossover and mutation, are applied over the individuals. Crossover is performed between two parent individuals to create offspring. Mutation is performed to randomly modify individuals by altering the genes. Once the genetic operators are applied, new generation is formed with the individuals that passes the fitness criterion. There are certain termination conditions, and, if they are satisfied, the process stops; otherwise, the process iterates to produce next generations. Termination conditions for genetic algorithm are:

- If a solution that optimally suits the problem is found
- If predefined number of generations are evolved
- Manual inspection
- Combination of all or any of the above

In the context of GANN model, genetic algorithm is used to train the neural network. It is used to find the network weights which would predict the best results and obtain a reasonable accuracy [25]. Neural network trained with gradient descent method has a tendency to converge at those weights which are not actually the best suitable weights for the network, whereas genetic algorithm due to its searching patterns searches the best suited weights for the neural network. To train the neural network by genetic algorithm, a weight set represents an individual which has been encoded as real numbers. The weight set contains all the weights and biases of network which are to be optimized. The fitness value of each individual can be calculated by the error obtained between actual output and desired output of the neural network using Eq. 11.7.

$$\text{fit}(i) = \frac{1}{\text{mse}(i)} \tag{11.7}$$

where fit*(i)* represents the fitness value of individual i and mse*(i)* represents the mean square error which is obtained by applying the weights of individual i to artificial neural network. Initially a population of individuals is generated using random numbers that lie within a specific range. Now genetic algorithm is implemented in three core steps: selection, mating, and next generation. Selection process selects an individual based on some criterion as defined by the algorithm [26]. There are several criteria to select an individual, e.g., roulette wheel selection, rank selection and tournament selection, etc. Here roulette wheel selection is used to select the variables. Roulette wheel selection is dependent on the probability each individual gets based on its fitness value. The probability p_k of each individual k is calculated by Eq. 11.8.

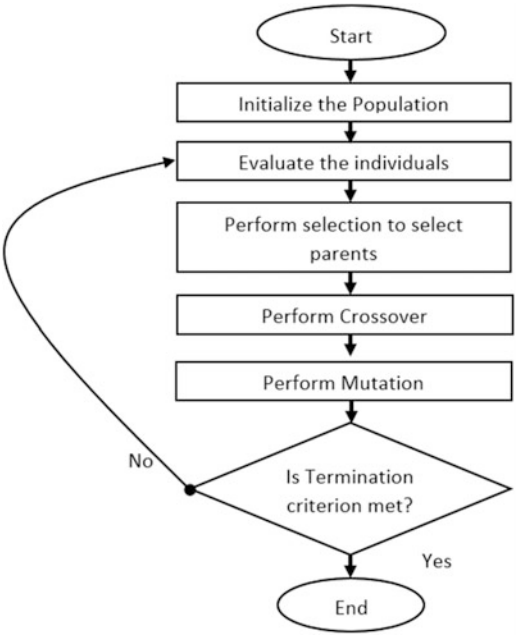


Fig. 11.5 Flow chart of genetic algorithm

$$p_k = \frac{\text{fit}(k)}{\sum_{i=1}^{m} \text{fit}(i)} \tag{11.8}$$

where fit*(k)* is the fitness value of individual k. Now each individual is given a range according to its probability. A random number is used for the selection of an individual. If the random number lies within the range of a particular individual, then that individual is selected; otherwise not. Once two parent individuals i and j are selected, they are ready for mating purpose. Mating defines two processes crossover and mutation. Crossover between the parents is done using the pseudocode of module *crossover_genetic_algorithm*().

```
crossover_genetic_algorithm(individual(i), individual(j))
offspring(1) = individual(i) * (1 - rand) + individual(j) * rand
offspring(2) = individual(j) * (1 - rand) + individual(i) * rand
```

where rand is a random number between 0 and 1.

Mutation is done to introduce random modifications to offspring i. Mutation can be done using pseudocode of module *mutation_genetic_algorithm()*.

```
mutation_genetic_algorithm(offspring(i))
if r and1< mcf
     offspring(i) = offspring(i) + offspring(i) * (u – l) * fg
```

where l and u are lower bound and upper bound that define the range of individuals, *rand1* is a random number between 0 and 1, *mcf* is mutation control factor, and fg can be calculated by Eq. 11.9.

$$fg = rand2 * \left(1 - \frac{g}{Gmax}\right) \tag{11.9}$$

where *rand2* is a random number between 0 and 1, g represents the current generation, and *Gmax* represents maximum number of generations. Having crossover and mutation applied successfully, a new population of individuals is obtained. The individuals in new population may replace parent individuals if they are better than parents. Their fitness value can be calculated using fitness formula described by Eq. 11.7. The higher the fitness value, the better the individual is. When a new generation is achieved, the entire process can be repeated until the termination criteria are not satisfied. Termination criteria can be one of the reasons as stated before. The entire process is described the pseudocode of module *gann()*.

```
gann()
Initial Parameters
    Initialize various parameters such as, population size (n),
initial population (pop), number of generations (Gmax), mutation
control factor (mcf), lower and upper bound of random numbers of a
candidate solution (l and u respectively)
Do
    for i=1 to n
          Calculate fit (i) using Eq. 7
    end
    for i = 1 to n
       Calculate p_k using Eq. 8
    end
    for i = 1 to n
          Apply roulette wheel selection to select two parents j
     and k using probabilities p_j and p_k
          crossover_genetic_algorithm(individual(j), indivi-
      dual(k)) using module 3.2.1
    end
```

(continued)

```
    for i = 1 to n
          Select a random number, r
          If r < mcf
             Mutation_genetic_algorithm(offspring(i)) using
          module 3.2.2
    end
    Replace older generation by new generation based on the fitness
    value of individuals
Until termination condition is satisfied
```

11.3.3 BMANN: Bird Mating Algorithm-Based Neural Network

BMO algorithm is inspired by the evolution process of birds and tries to simulate the process to optimize the solution. BMO has been proved efficient because it exploits distinct moving patterns to search in search space [19]. Mating process in birds is very similar to an optimization problem. In mating process, birds try to breed with another bird that has best quality genes, because birds with best quality genes live longer. During mating season, several features of birds such as singing, dancing, beak, tail, wing, etc. are considered. To produce a brood of higher quality, different strategies are employed by birds. There are four strategies which are employed by birds: monogamy, polygyny, polyandry, and promiscuity. These are:

- Monogamous bird: It refers to a male bird that mates with only one female bird.
- Polygynous bird: It refers to a male bird that mates with several female birds.
- Polyandrous bird: It refers to a female bird that mates with several male birds.
- Promiscuous bird: It is a mating strategy with unstable relationship, and the male bird will never see his brood and female bird.

Now the above four strategies can be simulated to solve an optimization problem. BMO algorithm is an iterative process which is shown in Fig. 11.6 and generates a population of individuals in each iteration where each population is called a society and each individual is referred to as a bird. There are four types of individuals (birds) in the society – monogamous, polygynous, polyandrous, and promiscuous – which are used to find an optimal solution. In this algorithm X denotes a set of birds where $X = a \cup b \cup c \cup d$ and a, b, c, d represents a set of monogamous birds, polygynous birds, polyandrous birds, and promiscuous birds, respectively. The gene vector of each bird is represented as $x(X) = (x(X,1), x(X,2), \ldots, x(X,n))$. For reducing the complexity, let us assume that only brood is produced when mating happens between the birds. This iterative process continues to breed among the society until maximum number of generations is produced.

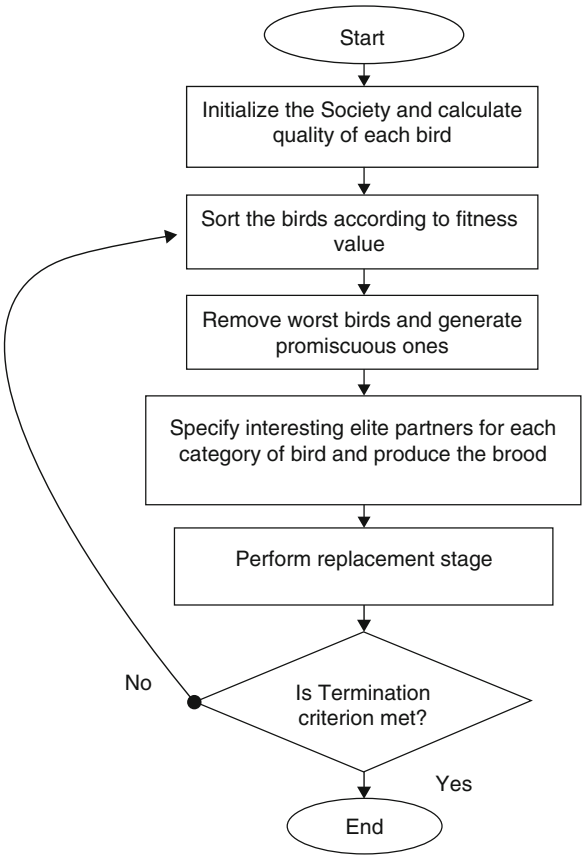


Fig. 11.6 Flow chart of bird mating algorithm

In this algorithm, it is proposed that the birds can change their types in different iterations according to their fitness value. The birds which have highest fitness value are chosen as polyandrous birds. The birds which have worst fitness value are selected and removed from the new generation and replaced by new birds. These new birds are referred to as promiscuous birds. Now the birds that have second best fitness value are chosen as monogamous birds, whereas remaining birds are considered in the category of polyandrous birds. Monogamous birds have fitness value better than polyandrous birds. The percentage of four types of birds is determined manually.

In the context of BMANN model, bird mating algorithm is used to train the neural network. Each bird represents a weight set, which consists of weights of the connections of neural network. The weight set is encoded using real numbers. A society of different weight sets (birds) is created which consists of four types of birds: monogamous, polygynous, polyandrous, and promiscuous. These birds are specified using the value of fitness function. Fitness function for this approach is defined by Eq. 11.7. The birds with worst fitness value are disposed, and new birds

are generated that are known as promiscuous birds. The birds with highest fitness value are categorized as polyandrous birds, the birds with second highest fitness value are referred to as monogamous, and the remaining birds are categorized as polygynous birds. In society of birds, monogamous and polygynous birds form the larger part of the society, whereas polyandrous and promiscuous birds form the smaller part of the society. Once the types of birds are specified, interesting elite partners for each kind of bird are specified.

Roulette wheel selection is used to obtain the interesting elite female birds for monogamous birds and interesting elite female birds for promiscuous birds. The better the value of fitness function, the higher is the probability of bird being selected for mating. Out of m birds, the selection probability of k^{th} bird is given by Eq. 11.10.

$$p_k = \frac{\text{fit}(k)}{\sum_{i=1}^{m} \text{fit}(i)} \tag{11.10}$$

where fit*(k)* is the fitness value of the bird k. Each bird is assigned a particular range based on the calculated selection probability. To select an interesting elite female bird, a random number is generated. Now the bird, whose range contains the random number, is selected as elite female bird for monogamous bird or promiscuous bird as the case may be.

Annealing function is used to obtain interesting elite female birds for polygynous birds and interesting elite male birds for polyandrous birds. Annealing function calculates the probability for polygynous and polyandrous birds to mate with their respective partners. The probability is calculated using Eq. 11.11.

$$Pr = \exp\left(\frac{-\Delta f}{T}\right) \tag{11.11}$$

where Pr defines the probability, Δf defines the absolute difference between the fitness value of bird under consideration and the corresponding partner, and T is the parameter to adjust the probability. Now a random number is generated; if that number is less than Pr, then the corresponding partner is selected; otherwise not. After the process of specifying the elite birds, mating is performed and fitness value is calculated.

Monogamous birds and promiscuous birds produce the brood in similar manner using pseudocode of module *monogamous_promiscuous()*.

```
monogamous_promiscuous(x(a), x(ef))
if rand1 < mut_ctr_fac
   x(brood) = x(a) + wt × rand2 × (x(ef) – x(a))
else
   x(brood) = x(a) + mut_w × ((rand3 - rand4) × (ub – lb))
```

where *rand1*, *rand2*, *rand3*, and *rand4* are random numbers, *wt* denotes weight that varies with time, mut_w is the weight for mutation, and *ub* and *lb* are upper and lower bounds on values of birds. *x*(*a*) represents the bird for mating and *x*(*ef*) represents its elite female partner.

Polygynous birds and polyandrous birds produce the brood in similar manner using pseudocode of module *polygynous_polyandrous()*.

```
polygynous_polyandrous(x(b), x(eb))
if rand1 < mut_ctr_fac
    x(brood) = x(b) + wt× ( n_{i=1} rand_i × (x(eb) – x(a))) / n
else
    x(brood) = x(b) + mut_w × ((rand2 – rand3) × (ub – lb))
```

where n is the number of birds required in mating for a polygynous or a polyandrous bird.

In the process of optimization, a defined percentage of birds with worst fitness value are discarded, and new birds, promiscuous birds, are generated using pseudocode of module *new_promiscuous()*. *x*(*b*) represents the bird for mating and x(eb) is a set of elite partner birds.

```
new_promiscuous()
x(d) = l + z^gen × (u – l)
z^(gen+1) = 4 × z^gen × (1 – z^gen)
```

where *z* is a variable whose initial value is generated randomly. This process continues until a maximum number of societies (generations) have been generated or an optimal set of weights has been obtained. The entire process is described the pseudocode of module *bmann()*.

```
bmann()
Initialization
    Initialize various parameters such as society size (n),
proportion of different birds, number of generations (max_gen),
mutation control factor (mut_ctr_fac) and other controlling
parameters
Do
    for i=1 to n
          Calculate fit(i) of each bird i, using Eq. 7
```

(continued)

```
    end
    Sort(birds) in decreasing order of fit(i)
    Categorize different types of birds as a, b, c, d as
    monogamous, polygynous, polyandrous and promiscuous birds
    respectively
    Discard birds(d), i.e., worst birds of the society and
    generate new set of birds, new_birds(d), using new_promisc-
    uous() module3.3.3
    for i=1 to d
          calculate fit(new_birds)
    end
    for i to a
           Apply roulette wheel selection to select an interesting
          elite female bird, bird(ef) using Eq. 10.
           monogamous_promiscuous(bird(i), bird(ef)) using
          module 3.3.1
    end
    for i to b
           Apply annealing function to select a set of interesting
          elite female birds, bird_set, using Eq.11.
           polygynous_polyandrous(bird(i), bird_set) using
          module 3.3.2
    end
    for i to c
           Apply annealing function to select a set of interesting
          elite male birds, bird_set, using Eq. 11.
           polygynous_polyandrous(bird(i), bird_set) using
          module 3.3.2
    end
    for i to d
           Apply roulette wheel selection to select an interesting
          elite female bird, bird(ef) using Eq. 10.
           monogamous_promiscuous(bird(i), bird(ef)) using
          module 3.3.1
    end
    Replace older society by new society based on their fitness value
    Update different parameters accordingly
Until termination condition is satisfied
```

11.4 Experimental Setup

The entire process of the proposed prediction model is simulated in MATLAB. Complete process starts with extracting features from the datasets. Two models for predictions GANN and BMANN have been used and implemented. Overall process is implemented in three major steps:

1. Selecting relevant features with the help of genetic algorithm
2. Implementing GANN, genetic algorithm to train artificial neural network
3. Implementing BMANN, bird mating optimizer algorithm to train artificial neural network

11.4.1 Dataset

Promise Software Engineering Repository provides five different datasets for the purpose of software defect prediction model. These datasets are created by NASA MDP (metrics data program) and are made available publicly [3]. These datasets contain the data of software which are written in different programming languages such as C and C++. The five datasets are as follows:

- **CM1:** It is the dataset which contains data of the functions written in C. It has 498 instances attributed by 21 metrics, out of which 449 instances represent "false" class and 49 instances represent "true" class.
- **JM1:** It is the dataset which contains data of the functions written in C. It has 10,885 instances attributed by 21 metrics, out of which 2106 instances represent "false" class and 8779 instances represent "true" class.
- **KC1:** It is the dataset which contains data of the functions written in C++. It has 2109 instances attributed by 21 metrics, out of which 326 instances represent "yes" class and 1783 instances represent "no" class.
- **KC2:** It is the dataset which contains data of the functions written in C++. It has 522 instances attributed by 21 metrics, out of which 105 instances represent "yes" class and 415 instances represent "no" class.
- **PC1:** It is the dataset which contains data of the functions written in C. It has 1109 instances attributed by 21 metrics, out of which 77 instances represent "false" class and 1032 instances represent "true" class.

The dataset contains 21 attributes which are categorized as 5 different lines of code, 3 McCabe metrics, 4 base Halstead metrics, 8 derived Halstead metrics, 1 branch count, and 1 class label as shown in Fig. 11.7.

The datasets which contain McCabe and Halstead metrics only are able to predict the error proneness of the software system because of the arguments presented by McCabe and Halstead. McCabe argued that the codes which have complex pathways are more error prone, i.e., codes with complex structure have more tendency to contain errors. Halstead argued that the code which is not

loc	Mccabe's line of code
v(g)	Cyclomatic complexity
ev(g)	Essential complexity
iv(g)	Design Complexity
n	Total operators + operands
v	Volume
l	Program length
q	Difficulty
i	Intelligence
e	Effort
b	Bug
t	Time estimator
locode	Line count
locomment	Count of lines of comments
loblank	Count of blank lines
locodeandcomment	Count of line of code and comments
uniq_op	Unique operators
uniq_opnd	Unique operands
total_op	Total operators
total_opnd	Total operands
branchcount	Branch count of flow graph
defects	{false, true}

Fig. 11.7 Dataset and its attributes

properly readable or hard to read has more tendency to contain errors. So based on these two arguments, McCabe and Halstead metrics are able to provide the information about whether the software might contain errors or not.

Each dataset is divided into two parts – training dataset (70%) and testing dataset (30%). The training dataset is used to build the model. The feature selection module is applied over training dataset, which gives five relevant attributes as result. The data of these attributes are extracted from training dataset and are fed into GANN module and BMANN module to perform training. The testing dataset along with optimal weights is applied to both the modules, and accuracy is measured once the optimal weights of neural network are obtained. Receiver operating characteristic curve is plotted to analyze the performance of both the models.

11.4.2 Genetic Algorithm-Based Feature Selection

Genetic algorithm-based feature selection employs the calculation of mutual information between different attributes internally. This process accepts feature_number, i.e., number of features to be selected. Several other parameters which are required for the algorithm are as follows:

- Feature_number = 5
- Individual_size = feature_number
- Population_size = 200 * feature_number
- Elite = 1
- Max_gen = 80
- Calculation of *Hx*, *Hy*, *MIxy*, and *MIxx*

After successful completion of the process, it gives a set of numbers which represent the extracted features. In context of this work, five features are selected. Data belonging to these features can be fed into neural network which has five neurons in its input layer. The output layer of neural network contains only one neuron which is used for prediction.

11.4.3 GANN: Genetic Algorithm-Based Neural Network

An artificial neural network is constructed in MATLAB which contains three layers, input layer with five neurons, hidden layer with ten neurons, and an output layer with only one neuron. Log-sigmoid (logsig) is used at both layers as the transfer function for calculating the output of each neuron. The connections between the neurons bear some weight value, and threshold is applied to neurons. There are 71 values of weights and threshold which are to be optimized using genetic algorithm. To set up the genetic algorithm, several parameters are required:

- Individual_size = 71
- Population_size = 100
- Weights are searched in a range of [−5, 5]: l = −5, u = 5
- mcf = 0.001
- max_gen = 50

After successful completion of this process, the optimal set of weights and thresholds is achieved. But being a stochastic process, the algorithm does not produce an accurate solution every time it executes. So, the algorithm is run 100 times to generate 100 optimal solutions. A best solution out of 100 solutions is selected as the weight set of the neural network.

11.4.4 BMANN: Bird Mating Algorithm-Based Neural Network

Similar to the above process, an artificial neural network is constructed with three layers where the input layer contains five neurons, the hidden layer contains ten neurons, and the output layer contains only one neuron. Logsig is applied to the neurons. This algorithm contains a society of birds, where each bird consists of a

vector of 71 values, where each value corresponds to a weight or bias of the network. To set up the bird mating optimizer algorithm, several parameters are required:

- Bird_size = 71
- Society_size = 100
- Monogamous birds = 50% of society
- Polygynous birds = 35% of society
- Polyandrous birds = 10% of society
- Promiscuous birds = 5% of society
- Weights are searched in a range of [−5, 5]: l=−5, u=5
- T, wt, and mut_w are linearly decreasing functions with initial values: T= [300, 50], wt= [2.5, 0.5], and mut_w= [0.1, 0.0001]
- mut_ctr_fac = 0.9
- max_gen = 50

After successful completion of this process, an optimal set of weights and biases is achieved, which can be applied to neural network for prediction. The entire process can be run over 100 times to improve the accuracy.

11.5 Analysis of GANN and BMANN

Once GANN and BMANN models are implemented and their results are obtained, receiver operating characteristic (ROC) curve is plotted for each dataset to analyze the performance of both models. But before implementing GANN and BMANN, there was a preprocessing task of selecting relevant features from the given set of features. The model selects five attributes from each dataset. Table 11.1 shows the attributes that are selected from different datasets.

11.5.1 ROC Curves

11.5.1.1 CM1

ROC curve for dataset "CM1" is shown in Fig. 11.8, which shows the performance of both the models. Curves clearly show that the performance of BMANN is better than GANN. In ROC curve analysis, GANN shows greater false-positive rate than BMANN which is not suitable to be accepted.

Table 11.1 Relevant feature for each dataset

Dataset	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5
CM1	iv(g)	l	e	locode	locodeandcomment
JM1	iv(g)	l	e	t	locodeandcomment
KC1	ev(g)	l	e	t	locodeandcomment
KC2	ev(g)	l	e	t	locodeandcomment
PC1	ev(g)	l	e	t	locodeandcomment

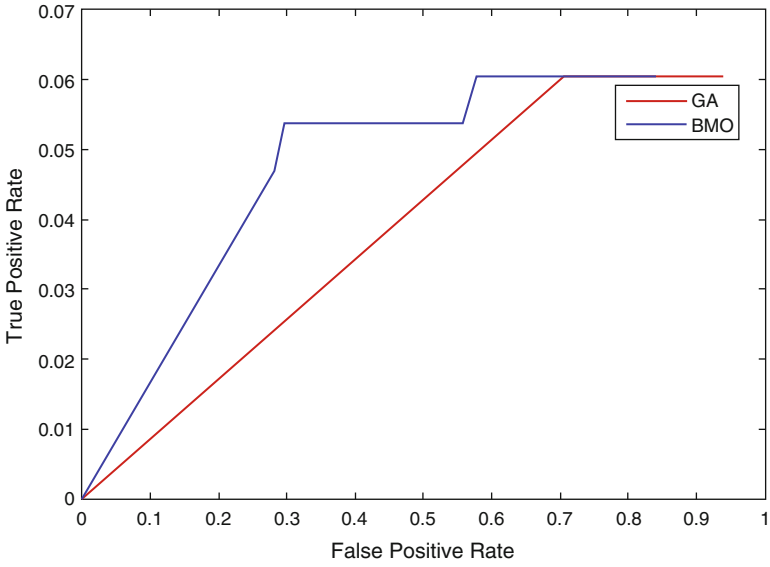


Fig. 11.8 ROC for dataset "CM1"

11.5.1.2 JM1

ROC curve for dataset "JM1" is shown in Fig. 11.9, which shows the performance of both the models. Curves of this dataset do not show a significant advantage of BMANN over GANN, but according to actual values BMANN performs better than GANN. In ROC curve analysis, BMANN reaches to a higher true positive rate than GANN, which is desirable.

11.5.1.3 KC1

ROC curve for dataset "KC1" is shown in Fig. 11.10, which shows the performance of both the models. Curves clearly show that the performance of BMANN is better

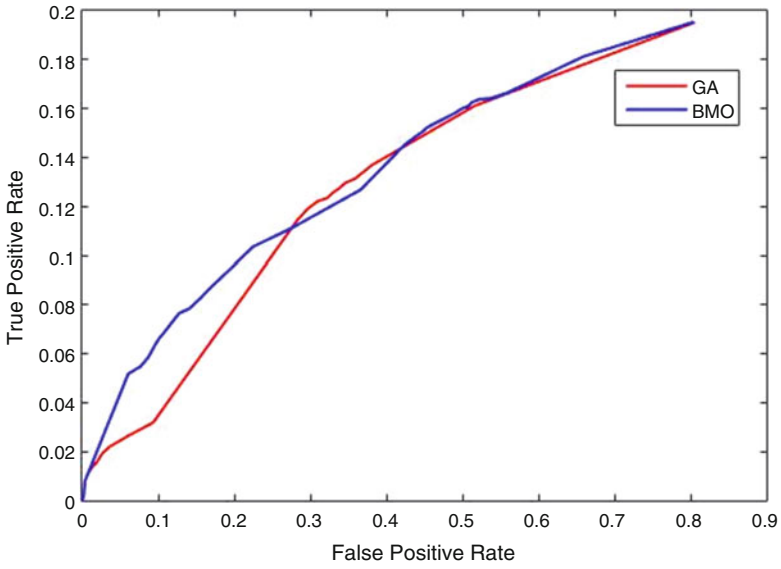


Fig. 11.9 ROC for dataset "JM1"

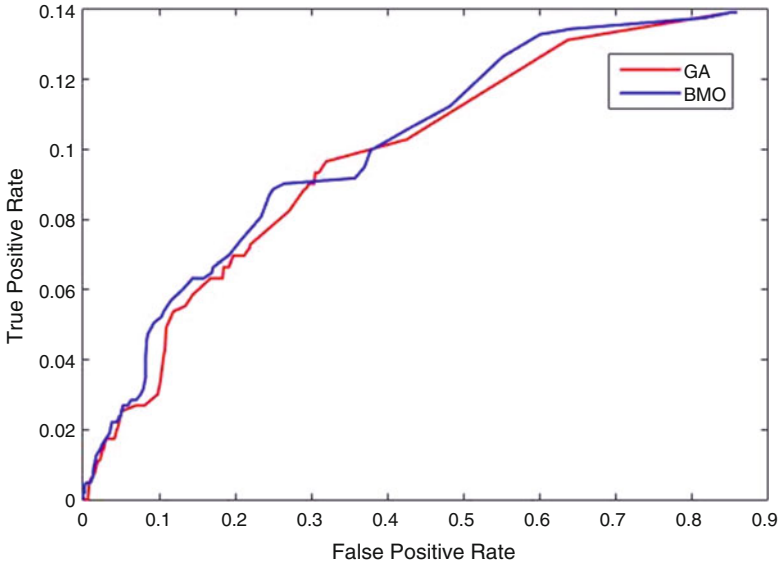


Fig. 11.10 ROC for dataset "KC1"

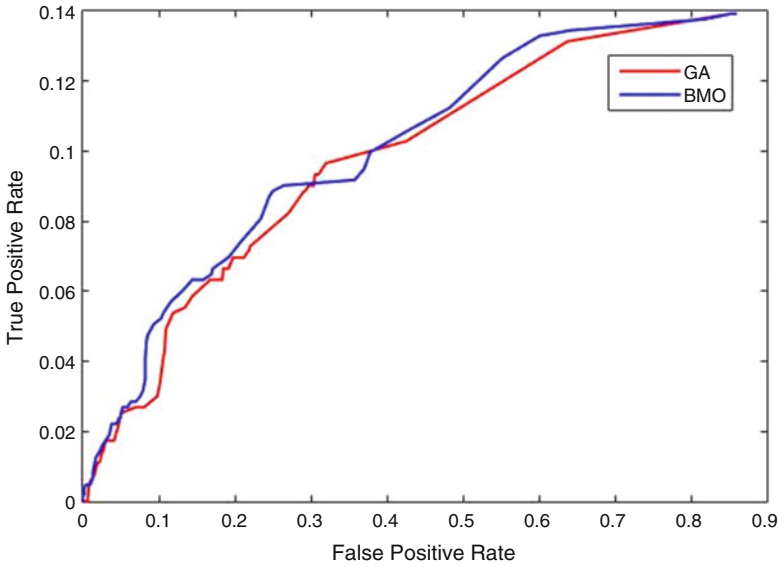


Fig. 11.11 ROC for dataset "KC2"

than GANN. In ROC curve analysis, BMANN shows desirable results, i.e., it reaches to a higher true positive rate than GANN.

11.5.1.4 KC2

ROC curve for dataset "KC2" is shown in Fig. 11.11, which shows the performance of both the models. Curves clearly show that the performance of BMANN is much better than GANN. In ROC curve analysis, GANN shows greater false-positive rate than BMANN, but on the other hand it reaches higher true positive rate too.

11.5.1.5 PC1

ROC curve for dataset "PC1" is shown in Fig. 11.12, which shows the performance of both the models. Curves of this dataset do not show a significant advantage of BMANN over GANN, but according to actual values BMANN performs better than GANN. In ROC curve analysis, GANN shows greater true positive rate than BMANN, but its false-positive rate is also higher than BMANN, which makes it less significant than BMANN.

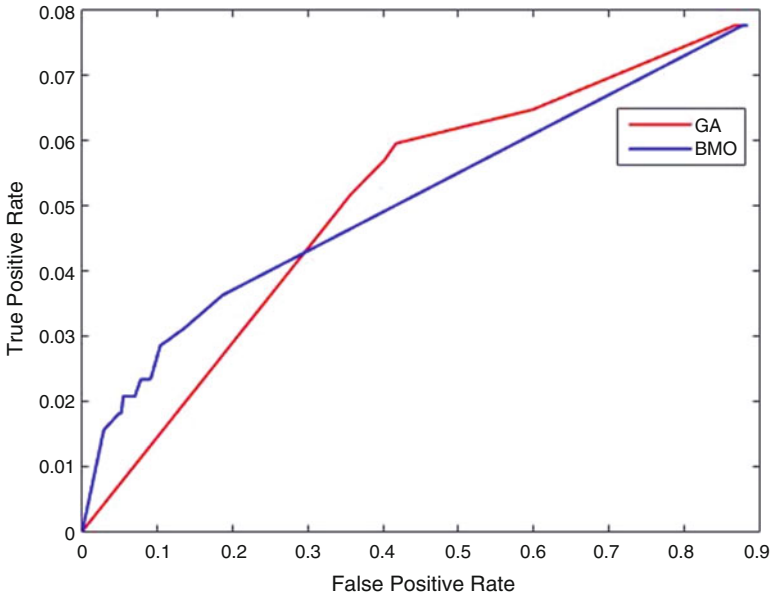


Fig. 11.12 ROC for dataset "PC1"

11.5.2 Accuracy

Accuracy of both models (GANN and BMANN) for every dataset is analyzed using ROC. Table 11.2 shows the accuracy (%) of GANN and BMANN for every dataset.

Comparison of their accuracy can be analyzed through the bar graph, which is shown in Fig. 11.13.

The comparison clearly shows that the accuracy of BMANN is better than GANN for each dataset. Bird mating optimizer algorithm produces better results as compared to genetic algorithm in the process of software error proneness prediction.

11.6 Conclusion and Future Work

This work attempts to emphasize the error proneness of software. Based on the value of different software matrices and past experience, it can be predicted whether the software is error prone or not. The prediction might help a lot of engineers to reduce the overhead of testing and produce high-quality software products. A lot of work has been done in this area, but this article attempts to improvise the performance of prediction using the hybrid approach of artificial neural network and bird mating optimizer algorithm. The results which are

Table 11.2 Accuracy of GANN and BMANN

Dataset	GANN (%)	BMANN (%)
CM1	87.24	93.95
JM1	76.72	80.88
KC1	82.27	86.23
KC2	75.64	80.76
PC1	88.34	91.96

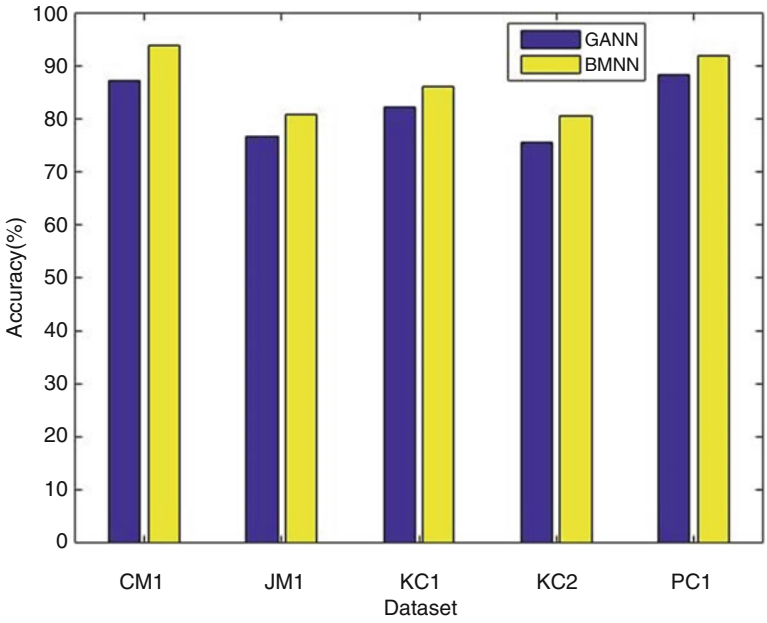


Fig. 11.13 Accuracy comparison of GANN and BMANN

produced from five different datasets, provided by NASA Promise repository, show that bird mating optimizer is efficient with respect to genetic algorithm when applied to the training process of artificial neural network. BMANN gives more accurate results than GANN. BMANN works efficiently than GANN. It might be because BMO in BMANN has more exhaustive and more rigorous searching patterns than GA.

11.6.1 Future Work

GANN and BMANN both train the neural network for prediction of software error proneness. This work can be extended to improve the parameters of BMANN. The parameters that are required in BMANN are decided using error and trial method,

but these parameters can be set using a definite method. The performance of BMANN can be further improved by optimizing the model parameters. Before applying BMANN to the dataset, the dataset can be preprocessed to produce more effective results. The emphasis can be given to improve the accuracy of GANN and BMANN in the future.

References

1. Ghezzi C, Jazayeri M, Mandrioli D (2002) Fundamentals of software engineering. Prentice Hall PTR, Upper Saddle River
2. Sandhu PS, Goel R, Brar AS, Kaur J, Anand S (2010) A model for early prediction of faults in software systems. In: Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on, vol 4. IEEE, pp 281–285
3. NASA (2005) PROMISE DATASETS PAGE. Retrieved November 4, 2016, from http://promise.site.uottawa.ca/SERepository/datasets-page.html
4. Board I (1993) IEEE standard classification for software anomalies. IEEE Std, 1044
5. Rumelhart DE, Hinton GE, Williams RJ (1985) Learning internal representations by error propagation (No. ICS-8506). CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE
6. Jacobs RA (1988) Increased rates of convergence through learning rate adaptation. Neural Netw 1(4):295–307
7. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier
8. Convolutional neural networks for visual recognition. Retrieved November 4, 2016, from http://cs231n.github.io/neural-networks-1/
9. Nawi NM, Ghazali R, Salleh MNM (1995) An approach to improve back-propagation algorithm by using adaptive gain. Off J Biomed Fuzzy Syst Assoc 125
10. Perez S (2008) Apply genetic algorithm to the learning phase of a neural network
11. Porter AA, Selby RW (1990) Empirically guided software development using metric-based classification trees. IEEE Softw 7(2):46–54
12. Askari MM, Bardsiri VK (2014) Software defect prediction using a high performance neural network. Int J Soft Eng Appl 8(12):177–188
13. Jindal R, Malhotra R, Jain A (2014) Software defect prediction using neural networks. In: Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2014 3rd International Conference on. IEEE, pp 1–6
14. Gayathri M, Sudha A (2014) Software defect prediction system using multilayer perceptron neural network with data mining. Int J Recent Technol Eng 3:54–59
15. Singh M, Salaria DS (2013) Software defect prediction tool based on neural network. Int J Comput Appl 70(22):22–28
16. Wahono RS, Herman NS, Ahmad S (2014) Neural network parameter optimization based on genetic algorithm for software defect prediction. Adv Sci Lett 20(10–11):1951–1955
17. Zheng J (2010) Cost-sensitive boosting neural networks for software defect prediction. Expert Syst Appl 37(6):4537–4543
18. Arar ÖF, Ayan K (2015) Software defect prediction using cost-sensitive neural network. Appl Soft Comput 33:263–277
19. Askarzadeh A, Rezazadeh A (2013) Artificial neural network training using a new efficient optimization algorithm. Appl Soft Comput 13(2):1206–1213
20. Liu H, Motoda H (2012) Feature selection for knowledge discovery and data mining, vol 454. Springer, New York

21. Peng H, Long F, Ding C (2005) Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell 27(8):1226–1238
22. Ludwig O, Nunes U (2010) Novel maximum-margin training algorithms for supervised neural networks. IEEE Trans Neural Netw 21(6):972–984
23. Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85
24. Obitko M (1998) Main page – introduction to genetic algorithms – tutorial with interactive java applets. Retrieved November 4, 2016, from http://www.obitko.com/tutorials/genetic-algorithms/
25. Leung FHF, Lam HK, Ling SH, Tam PKS (2003) Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Trans Neural Netw 14(1):79–88
26. Jiang J (2013) BP neural network algorithm optimized by genetic algorithm and its simulation. Int J Comput Sci 10:1

Chapter 12
Improved Agile: A Customized Scrum Process for Project Management in Defense and Security

Luigi Benedicenti, Paolo Ciancarini, Franco Cotugno, Angelo Messina, Alberto Sillitti, and Giancarlo Succi

12.1 Introduction and Motivation

Developing software has become an increasingly more complex endeavor, especially in well-established domains with rich semantics and multiple, often conflicting needs coming from multiple customers. In the armed forces domain, these factors have traditionally been addressed via rigid development processes. Such processes seemed to conform well to the highly hierarchical structure believed to be integral part of a military organization. But today this is no longer the case.

Asymmetric opponents, multiple conflicting needs, and the imperative for a high-quality, flexible, and agile response have changed military doctrine profoundly. Although the effect of multiple, rapidly changing requirements coming from different actors had been identified for a relatively long time – Clausewitz was arguably the first to systematize the study of this effect and called it "friction" [1] – only recently has it come to bear on the creation of complex support tools like command and control systems or logistic management systems.

L. Benedicenti
University of Regina, Regina, Canada
e-mail: luigi.benedicenti@uregina.ca

P. Ciancarini
University of Bologna, Bologna, Italy
e-mail: paolo.ciancarini@unibo.it

F. Cotugno
Italian Army General Staff, Rome, Italy
e-mail: franco.cotugno@esercito.difesa.it

A. Messina • A. Sillitti (✉) • G. Succi
Innopolis University, Innopolis, Russian Federation
e-mail: a.messina@innopolis.ru; a.sillitti@innopolis.ru; g.succi@innopolis.ru

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_12

In addition to these considerations, new restraints in the defense budgets have effected a deep cultural change in the armed forces. For example, the US Department of Defense has changed its procurement policies to allow for common off-the-shelf equipment to be sourced instead of specialized equipment. Software production, however, is changing less rapidly, due in part to the absence of reliable methods to deal with complexity and rapid change in an already well-structured environment.

On the other hand, software cost per executable line of code (ELOC) or function point has decreased significantly in the commercial world in the last decade due to effectiveness of the latest generation of programming languages and the availability of open-source computer-aided software engineering (CASE) tools. Moreover, the adoption of agile development methods appears to have contributed to increased software development effectiveness and efficiency in those contexts.

At the General Staff of the Italian Army, these factors have led to the internalization of its software development, which requires contractors to work with a mix of civilian and military development teams. In turn, this has led to the creation of a new software development process, based on agile principles, and more specifically on scrum, called iAgile (for improved Agile).

Project management is an integral part of this process, because even in an agile context, it is still necessary to respond to the needs of the armed forces, chief among which are security and customer satisfaction. Drawing on the principles of agile development and intersecting them with the Italian Army procedures resulted in the incorporation of the following aspects in the management strategy:

- Frequent releases.
- Change management.
- Effective cooperation in a complex environment between requirement owners (goal owners), acquisition owners (gold owners), and contractors, who are numerous in the defense sector and do not traditionally interact easily with one another.
- Focus on small and empowered teams, to be able to react fast to evolving customer needs and emergency situations

The infrastructure of the iAgile process consists of four main “pillars”: agile training, innovative CASE tools, structured user community governance, and custom agile development doctrine. Refer to Fig. 12.1. All these pillars are part of the management strategy that enables iAgile to be effective.

Agile training is fundamental for change management, as it builds the knowledge needed to operate within the iAgile framework. But it is not enough, because change requires both motivation and reinforcement to be actualized. Motivation is provided through standard organizational means at the beginning, but after the first sprint, it is fueled by the visible progress demonstrated by the functionality of the partial product. Reinforcement is provided via the structured user community governance that enables feedback to be immediate and in bringing users close to the development greatly increases user satisfaction and consequently the team’s level of confidence.

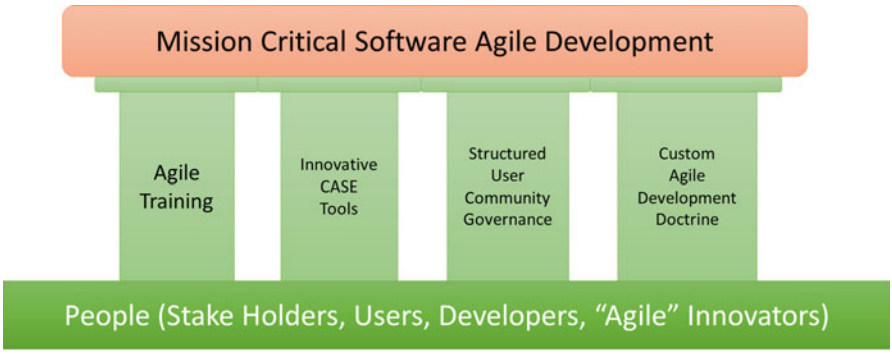


Fig. 12.1 Infrastructure Supporting Agile Software Development

The innovative CASE tools pillar is needed because without supporting tools tailored for iAgile, the software development would proceed at a slower pace due to lack of automation and a progress monitoring system. In particular, it was essential to develop a suite of noninvasive monitoring tools that measure the software artifacts being created without requiring developer intervention, which would slow down developers. The monitoring tools could then generate progress reports that concretely demonstrate the effectiveness of the development, acting as a reinforcement mechanism and, when needed, providing data for causal analysis and problem resolution.

The structured user community governance is needed to empower selected experts to act as delegated product owners with the same decision power as their department heads. Achieving this step required the full support of the top-level authorities in the Army. This generated the required level of autonomy in development teams so that development could proceed without the constant bottleneck of top-down authorizations. It was possible to realize this structure because the unit of development in our case is a single user story, which can be altered very rapidly and thus is perceived to have little impact on the project; yet, the freedom to develop a large number of user stories concurrently results in a marked productivity increase.

Last but not least, a doctrine had to be developed to ensure that any replica of the process throughout the military environment be coherent with the iAgile process methodology.

Naturally, it is important to build a management team suitable for such process. As the development effort grew, we found it necessary to create more teams and link the teams together using a variant of the "Scrum of Scrums" method. The development began with a single seven-people team and ended up with seven teams similarly construed. This led us to the lengthening of our cycles to 5 weeks from an initial duration of 3 weeks, to accommodate the interaction among the groups. It also called for the redefinition of the roles of product owner and scrum master. The product owner role had the added responsibility to keep track of the stakeholders. This "wayfinding" role was not explicit in our process initially, but given the number of stakeholders involved in the project, it was a necessity. The scrum

master role was redefined in terms of hierarchical authority. This was needed to make it possible for the scrum master to have the authority to resolve the issues brought forth by team members. Additionally, the scrum master also actively elicited feedback from all team members, rather than waiting for feedback passively. As scrum masters gained authority, it also became necessary to assign them the responsibility to keep the feedback loop running. To help enact the Scrum of Scrums, we then created the role of global product owner. This role became responsible for the synchronization of each individual scrum. Finally, as the skill sets grew, it became necessary to create a new role that reflects the awareness of the skill sets and their location in the teams to plan for skills allocation in the next sprint. We called it the scrum coach.

These management structures and role changes deviate from the scrum method and differentiate iAgile from standard scrum, justifying also its continued success since its very first project.

To test this project management strategy, we developed an entire command and control system for the Italian Army. This development was the first application of iAgile, and as a result it is fair to say that iAgile evolved concurrently with the development of our first case study. In this chapter, we will show how this development project evolved and the lessons we learned by running this project. The results will show that iAgile has been able to achieve high levels of productivity, quality, and customer satisfaction, but this has come at the cost of a profound cultural change in the development teams.

This chapter is organized as follows: The next section contains the state of the art on agile management. The next three sections provide a description of the method we developed together with the tools and the anticipated benefits of adopting our method. Section 12.6 presents a case study for the application of our method to build a command and control system. Section 12.7 contains the conclusions and future work.

12.2 State of the Art

12.2.1 Agile Project Management

Conventional project management is a storied discipline. Its roots can be traced back to the early 1900s through the contribution of pioneers such as Henry Gantt and Henri Fayol. Modern project management began in the late 1960s, and today there are numerous certifications for project managers, perhaps the most important of which are supported by the International Project Management Association and the Project Management Institute. In the late 1980s, the manufacturing industry began to change project management practices to include ways to increase efficiency and reduce semiworked product stock through the practice of just-in-time

manufacturing which, in turn, led to innovative project management methods like kanban [2] and lean [3].

The principles and practices of project management, usually, rely on the relative predictability of the scope of the project and on a set of preset requirements. Often, requirements are somehow formalized, and if there is a business agreement among partners, requirements are usually included in it. However, this method creates gaps. These gaps are of three kinds: The first kind is the gap between desired outcomes and actual outcomes. This originates from our inability to act precisely on a rapidly changing external environment, and it is compounded from the lack of perfect information on this environment. The second kind of gap is the gap between plans and actions in organizations. This originates from the inherent difficulty of defining a course of action for every single possible situation that might occur, which generates inability to achieve the planned outcome. The third kind of gap is the gap between actions and outcomes. This originates from the inability to foresee how actions will generate outcomes given a changing environment [1].

To close these gaps, a new type of project management is needed: one that is more flexible and is able to be proactive toward rapid requirement changes. This kind of project management is inspired by the Agile Manifesto, which was originally conceived for software production, but is now more widespread in its applicability [4]. Thus, the first books on agile project management (e.g., [5]) try to summarize some of the concepts in agile management, but because there is not much experience in the field, the suggestions and analyses emphasize the differences between conventional and agile project management, which carries the danger of overlooking important lessons from the traditional project management discipline. For example, the role of upper management in agile projects is to manage "the linkages between strategy and tactical project execution," while the traditional role is to approve projects and commit resources [5]. In reality, a more integrated approach in which both roles are adopted is better suited to a rapidly changing environment.

Most of the progress in agile project management was the result of experiential learning. For example, Jeff Sutherland et al. [6] documented the use of distributed scrum: a version of agile project management and a tool to increase productivity and team effectiveness.

In 2008, grounded theory proposed to investigate agile project management, which signals the continued interest in characterizing agile project management from actual practice [7]. In particular, areas of investigations involve the role of the project manager in an agile project, the process and problems of transitioning into an agile framework, and management of offshored or outsourced agile projects.

More recently, a systematic literature review on agile project management has been developed in 2014 by Chagas et al. [8], which contains information on 34 primary studies to create an ontology of terms and principles in the context of maturity models, to facilitate the transition between the two methods and consequent mentalities.

A better definition of agile project management has also been developed in [9]. According to this definition, agile project management deals with the

complexity and uncertainty of a software project, which creates unplanned change. To do that, agile project managers rely on some principles directly related to the ones expressed in the Agile Manifesto. Some of the practices in agile development methods, for example, team autonomy and continuous learning, are naturally extended to agile project management as well.

These principles have been expanded to more elaborate methods such as the Crowder and Friess agile project management [10]. Still, these ideas need additional validation and tailoring to specific situations. This is why case studies are of fundamental importance to understand how the principles formulated in a document can be instilled into an actual project and what lessons can be learned from the project's outcomes.

Currently, agile project management is entering the mainstream. For example, de Kort just published a book [11] on development and operations (DevOps) on the Microsoft stack that contains guidelines on how to use Microsoft tools in an agile context. This is but one of the many publications devoted to improving the experience of mainstream developers wishing to adopt a fully agile approach.

Yet, there are still some issues emerging from agile project management practices. One of them appears to be particularly important: communications. This issue occurs in all kinds of project management, not only the agile kind, but given that some of the standard tools for communication like project documentation are somewhat reduced in an agile approach, communication becomes especially relevant in an agile project. At the time of writing, there is no general solution for balancing communication and productivity, and this issue remains the subject of active research.

12.2.2 Project Management in Defense Domains

The most used project management methodology in the military environment is Projects in Controlled Environments, version 2 (PRINCE2® [13]), and its derivatives, e.g., North Atlantic Treaty Organization (NATO) [12]. This type of management activity requires heavily trained professionals and a massive quantity of formal documentation both in the planning and execution phases. For rapidly changing scenarios and volatile requirements, these articulated and time-consuming methods may be difficult to apply. PRINCE2® is a project management process-based approach method. Eight basic processes intended to be applied by a project manager in managing a project are modeled. The modeling is made in terms of steps in a logical sequence. PRINCE2® is adaptable to various types of projects in a different range of complexity. In the manual (Managing Successful Projects with PRINCE2®, 2009), a number of "components" are listed as guidance for a project manager in applying the process model. PRINCE2®, differently from other methodologies based on the availability of information about proven practices, provides a more prescriptive set of steps to be followed by project managers and teams. A possible advantage of PRINCE2® comes from the fact that it is

prescriptive in a large measure, and it may induce a degree of standardization in medium to large organizations. Tailoring of the methodology to specific projects is possible provided that the same basic steps are kept and the same terminology is used. Relevant benefits in corporate program management can be observed especially in the areas of staff training and project performance tracking. The possible constraints to creativity are a disadvantage especially in the area of software development where the human factors are becoming more and more relevant to the design solutions.

12.3 The iAgile Development Process

iAgile is a software development process based on distributed scrum [6]. Just like scrum, iAgile is an iterative development process in which a product owner, a scrum master, a manager, and a development team collaborate to deliver concrete functionality in a sequence of short development cycles (the sprints). However, the unique challenge of working within an armed forces context and having to include civilian consultants no longer as isolated subcontractors but as integral part of the development team led us to some changes that differentiate iAgile from scrum substantially. Furthermore, iAgile's domain is mission-critical software for the defense environment, which adds specific nonfunctional requirements like security and graceful system degradation. These requirements too contribute to further differentiate iAgile from standard scrum processes adopted in the enterprise.

In fact, iAgile has a number of formalized procedures that are assigned to personnel in various iAgile roles. Many of these roles are derived from the scrum method, but often there is a significant reshaping of the responsibilities associated with the roles. The basic principles of iAgile can be summarized as follows:

1. Development teams are the most relevant assets in the production process, but their activity has to be accompanied by a set of objective, real-time, noninvasive measures of effectiveness and performance agreed upon by team members and fully understandable by the customer. The set of measures must cover the most relevant aspects of the product as indicated by the user/stakeholder priorities (safety, security, quality, etc.). The first users of the measures are the team members themselves. The measures are collected and displayed in a software development virtual control room. Reports are automatically generated and sent to all the relevant community members.
2. The traditional role of product owner (PO) is shared by the global product owner board instead of being covered by a single individual. The GPO always includes a customer stakeholder representative, the most relevant application domain experts, and the teams POs. The team POs are responsible for team synchronization and feedback, which is an important feature to ensure the teams are always aligned in their production objectives and schedules.

3. The scrum master (SM) role in iAgile has been reinforced and has acquired a program management task. The SM has no decision power over the team member (in fact, scrum masters are themselves team members), but the reports the SM produces are the only means to assess the development status and the team performance. Every report is shared with the team members before reaching the GPOs and the stakeholders.
4. The skills of the technical team members are accurately scrutinized during the preliminary team building activities. The capability of working in parallel with subject matter experts not necessarily having a software engineering background is a key factor for selection of the team members. The developers are supposed to apply extended "pair programming" with asymmetric roles where the second programmer in the pair can be a security or quality expert. The technical growth of the team members is implemented throughout the whole production process and obtained by the insertion of "knowledge acquisition user stories" in the product backlog. Asymmetric pair programming in particular is not commonly used in scrum.
5. A new role was added to the process: scrum coach. This role is responsible for keeping track of the skills matrix in the development team and informing the asymmetric pair programming process. In standard scrum development, the team has no skill differentiation. In our environment, however, there are a variety of skills that coalesce. They come from the unique mix of consultants and armed forces personnel needed to be able to tackle the complexity of user requirements and the context within which the armed forces operate today. Not keeping track of the team's skill set would be dangerous because it could result in the misunderstanding of the tacit assumptions that often are an integral part of user stories.
6. A network of all relevant stakeholders (decision makers, top users, and application domain experts) is established at the project start and managed by the GPO as a professional social network. The network is used to share all the project-relevant information and is consulted when preparing all major project decisions.
7. All the scrum-derived "rituals" (stand-up meetings, sprint planning, reviews, and deliveries) are documented in electronic form or taped.
8. The reconciliation of the roles played in iAgile and the roles played as members of the armed forces needs to be addressed strongly and decisively through change management practices. These include, for example, a top-level commitment, in our case coming directly from the Army Chief of Staff; a clear sense of urgency, which in the Army is easier to instill; and a clear identification of the final outcome, which generates the motivation to achieve such outcome and also provides a clear, measurable target whose achievement can be detected and measured unequivocally [14]. This last point merits some further elaboration. Often, project managers define the final outcome in terms of percentage of implementation of the product's requirements and deviation from the planned budget. In our case, however, the outcome is defined in terms of costs, customer satisfaction, and quality. These project objectives are independent of the method

and tools used to manage the project and can be defined unequivocally while allowing uncertainty about requirements and the development process, which is the staple of agile development methods.

The principles articulated above contribute to create a very different process from the standard enterprise scrum. This creates a number of challenges, the most relevant of which are training and user community governance. Training is crucial for team members so that they can become productive team members immediately. Because iAgile is new and evolving, training evolves as well. It was thus natural to involve academia not only in the design of the process but also in the development of appropriate training programs for the various roles.

User community governance is a challenge because the concept itself, at the beginning, was nebulous for the armed forces. The challenge arises because of the large number of implicit assumptions when describing user stories in a highly specialized environment. Normally, this would not be a problem because the development team is familiar with that environment. However, it turns out that the assumptions are rarely checked for consistency; and when consultants are added to the environment, it becomes apparent that leaving these assumptions in the implicit state does not lead to successful projects. The following sections briefly describe how we approached both challenges.

12.3.1 Training

This is probably one of the areas containing the most significant differences when compared to traditional agile. iAgile is continuously changing to incorporate the awareness of potential vulnerabilities concerning mission-critical applications. Up-to-date knowledge on software engineering at the theoretical level is required to fully understand the true nature of the software product and the development of the production techniques. Depending on the specific role in the process, more focused training is needed, emphasizing realistic exercises and role playing.

All the components of the development teams (technical, management, and specific domain expertise) have to be familiar with the key concepts of software engineering and its evolution to fully understand the nature of the software product and be aware of the differences between the software development process and any other industrial product. This awareness has to be shared in the teams, and for this reason a common theoretical introductory part is included in every iAgile training course. A specific practical part depending on the role to be covered in the production process follows the first segment. Software engineers are trained on specific iAgile techniques such as requirement gathering and understanding and the sprint execution, whereas process managers are trained on noninvasive program management techniques.

12.3.2 User Community Governance

One of the key features of iAgile is the full involvement of the user community in the software development process. The area of mission-critical software in the Defense & Security area is characterized by substantial complexity of the user requirements often specified through a synthetic "mission need" document where most of the real needs are implicit or embedded. The user/stakeholder has relevant domain-specific expertise, which is very difficult to elicit but is pivotal for the correct implementation of the application. As reported in the case study, it is quite normal when the user begins to fully understand his or her own requirement only after several product deliveries. Besides the direct involvement of selected user experts in the development teams, in this environment, the realization of a network where a broader community of users/stakeholders can be reached is vital for the success of the iAgile project.

12.4 iAgile Support Tools

In iAgile, a number of noninvasive tools are used to monitor the rapid production cycles and report in real time on all necessary effectiveness indicators. Typical measures are code complexity (McCabe, Halstead, etc.), number of produced lines/programmer/day, number of bugs and fixes (code defect rate), number of user stories elaborated, user story quality estimation, US tracking, etc. Most of the measures are taken in real time and shared with the development teams using a simple presentation software product (software development control room).

Similar to all agile approaches, iAgile is based on the values of the Agile Manifesto and in particular on "Individuals and interactions over processes and tools" [15]. However, this does not mean that tools are not important in iAgile (or in any other agile approach); it means that tools should not create artificial constraints to how people work and collaborate, as they are only a support to perform activities in a more efficient way. In particular, tools should be easy to use and flexible enough to adapt to the preferred way of working adopted by the development team. Therefore, the tools originally developed to support traditional, plan-based development approaches can be hardly adapted to support the agile ones. There are a number of software vendors that propose their tools that have been designed to support traditional approaches with new versions and/or extensions to include also the agile approaches. However, such one-size-fits-all approach to support any development methodology adopted inside a company may create several problems to agile teams, in particular:

1. *Too many functionalities*: supporting a number of different development approaches forces tool vendors to include many features that can make the tools difficult to use and require a relevant amount of training. In particular, agile approaches usually require a limited amount of functionalities compared to

a traditional, plan-based approach; therefore, developers are easily lost in such environments and demand simpler tools that focus on their needs.
2. *Difficulty to configure*: due to the support of a wide range of processes, such tools require a complex customization activity before the tools can be actually used. Moreover, such configuration needs to be synchronized among all the installations that are usually local.
3. *Heavy tools*: the amount of functionalities offered is also connected to the systems requirements for the machines running such tools that may interfere with the development activities slowing down the developers' machines.
4. *Use of old paradigms*: most of such tools are based on old-style client-server architectures, require specific operating systems, and are not available through browsers or on mobile devices. This is a direct consequence of the evolution of the tools designed to support traditional, plan-based approaches that have been designed a long time ago, and vendors find their adaptation to the new paradigms difficult to implement.

Such limitations affect agile teams in a negative way, reducing their effectiveness in particular in the early stages of the introduction of an agile approach in a company. This is an important step: the team members need to develop in a very different way compared to the past, and the results of such early adoptions are often under strict scrutiny by the management. Such results may affect deeply how agile is perceived in the organization and whether the new methodology will be widely adopted in the company or dismissed as ineffective.

Even with all these limitations, this one-size-fits-all approach is considered the safest one in many organizations that are used to traditional development approaches for several reasons:

1. *Well-known tools*: the agile support has been added to tools already used in the organization providing at the beginning a certain level of confidence to both the management and the developers. However, in many cases, developers realize the limitations of such tools quite soon, dramatically reducing their use since they do not fit perfectly the process they want to adopt.
2. *Well-known vendors*: many providers of agile specific solutions are quite new to the market, and many large companies have never acquired software from them before. This is particularly critical when vendors are start-ups and small and medium enterprises (SMEs) that could disappear from the market at any time creating relevant problems to the users of their technologies. For this reasons, many companies prefer to adopt tools from large and well-known players with whom they have long-term relationships and they think they can trust. This is a relevant risk that is taken into consideration by many companies, especially the large ones. However, the increased use of open data formats can mitigate it and reduce the real risk of adopting tools from such start-ups and SMEs.
3. *Tools homogeneity*: introducing new tools in a company (especially in the large ones) requires a relevant amount of effort (e.g., installation of server and client components, system administration, training, etc.). For this reason, the tools used by the different team are often standardized, and the organizations prefer to

adopt for agile development the tools that are already well known. However, this may affect the agile teams as discussed before.

4. *Integration with already existing systems*: many project management and development support tools are deeply integrated with the information systems of a company, especially in the large ones. This integration provides several advantages in the day-to-day operations but creates a huge resistance to change. Moreover, in many cases, the processes are adapted to the existing infrastructure and not vice versa. This creates a number of problems for the flexibility of an organization and prevents them to experiment new approaches even if they can provide relevant benefits. Such deep integrations are not implemented anymore in modern architectures since the flexibility of the processes has acquired more relevance.
5. *Communication with the management*: monitoring the status of a project is of paramount importance for the management. This activity is often performed through standard reporting systems that are deeply integrated with the information systems of the company. Agile is completely different from this point of view, and standard reporting approaches are usually difficult to implement and are hardly adaptable to such approaches. Therefore, agile teams are often perceived as black boxes where it is difficult to expose the status of a project to other parts of the organization. For this reason, specific tools that fit the agile environment but are able to communicate with the management outside the team are needed to guarantee the success of agile approaches in many organizations.

12.4.1 Evolution of Tools

iAgile is designed to bring agile approaches inside mission-critical projects that are traditionally implemented through plan-based methodologies. Usually, such projects are carried out inside organizations with a quite rigid structure that consider agile approaches difficult to be implemented in their context. In this kind of context, regular agile development techniques cannot be adopted for several reasons, including the lack of specific tools designed to support the agile development in conjunction with the overall organization structure. Mission-critical projects often have strict requirements about providing elements that are needed by certification authorities. Even if certification standards do not prescribe the usage of a specific development approach, almost only the plan-based ones have been used in such contexts, and only in such areas support tools are available. However, the market needs (e.g., reduction of costs, improvement of customer satisfaction, volatility of requirements, etc.) are pushing companies to experiment and adopt novel development approaches. Therefore, the development of specific tools able to support the agile development considering the certification aspects is needed.

For a successful implementation of the iAgile methodology (and for many other agile methods), many support tools are required to speed up activities, share information easily, and assess continuously the status of a project [16]. However,

according to the values and the principles of the Agile Manifesto, the development team has to focus on delivering working software removing whatever is not valuable for the customer [17, 18]. Moreover, the same set of values and principles state that the development team should periodically reflect on how to become more effective and tune their behavior accordingly. This implies that a deep analysis of the effectiveness of the development team is performed analyzing different aspects of the work such as the effort spent and the quality of the source code, as such aspects are connected with the principles of the Agile Manifesto.

The implementation of a monitoring activity that inspects continuously the effort spent and the quality of the source code has been implemented in very different ways in the past following the guidelines identified by the personal software process (PSP) and the team software process (TSP) in conjunction with the requirements of the levels of the Capability Maturity Model (CMM). However, such approaches have several limitations including:

- They have been developed in an era when all development approaches were plan based (mainly waterfall, V-shape, and iterative).
- The data collection and analysis was only partially automated requiring a large amount of context switching between the productive activities and the monitoring ones. Such switches are known to create a number of problems interfering with the productive activities.

Therefore, the monitoring activities need to be implemented in a completely new way to be effective for an agile development team.

12.4.2 Noninvasive Measurement Tools

Noninvasive measurement techniques are the only approaches able to satisfy both the continuous monitoring needs and the strong focus on valuable activities for the customers. For this reason, a noninvasive measurement infrastructure is needed to support effectively the iAgile development process.

In particular, during the early phases of the definition of the iAgile methodology, several research prototypes have been adapted and tested in the team to verify the suitability and effectiveness of the approaches. Such prototypes were able to collect both process and product metrics. Process and product metrics are discussed below:

- *Process metrics*: such metrics describe characteristics of the development process investigating the amount of effort spent in specific activities and the role of each team member inside the development, where the knowledge is kept and how it is shared among the team members; how defects are collected, managed, and fixed; etc. Basically, process metrics provide an analysis (at different levels of detail) of the behavior of the team as a whole, of the behavior of each team member, and of the collaboration among team members [19–23]. Such analyses are useful to perform several activities including:

- *Providing a high-level overview of the status of the project*: data collected from each team member and from repositories (e.g., issue tracking systems, version control systems, etc.) can be integrated to provide a big picture of the overall status of the project and the development team identifying where the effort of the team is spent, how much of it is dedicated to the development of new features and how much to the maintenance of the already existing ones, etc. Moreover, this kind of analysis can provide information about how fast is the team in implementing user stories, the level of completion of the project, the level of volatility of the requirements, etc.
- *Providing feedback to each team member about its contribution*: a data collection and analysis program is effective only if the people involved get benefits from it. In particular, the implemented program is designed to provide detailed reports about the activities performed by each team member describing the activities performed at a fine-grain level to help them to improve their work habits and improve their overall effectiveness.
- *Accessing the level of collaboration inside the team*: the data collected is able to highlight how much collaboration is performed inside the team looking at the artifacts that are shared and are modified. This allows an indirect assessment of how people collaborate and how the knowledge about specific parts of the source code is shared across the team identifying areas that need improvements. In particular, it is possible to identify areas that are not covered enough, and there is a need of ad hoc training activities and/or knowledge sharing among team members.

Process metrics are extracted by different tools that are intended to collect different kinds of data. In particular, there are different kinds of process data that can be collected. We can classify them as metrics collectable online and metrics collectable off-line:

- *Online metrics collection*: the data collection system required the installation of plug-ins for the development environment to be able to monitor constantly the opened files and how much time each developer spent in each file, class, and method of the system under development. Moreover, another tool working in background was able to trace all the active applications and keep track of the time spent in each one. This tool was useful to trace the applications used other than the development environment. Such data need to be collected during the development since they are not collected by any other system. Therefore, a continuous data collection is important to avoid missing data.
- *Off-line metrics collection:* the data collected in this case are collected in an indirect way from a number of different kinds of artifacts such as bug reports and source code commits. In particular, data about time required to close issues and the related fix, who deal with them, etc. Such data can be collected at any time since it is stored as a side effect of storing other information (e.g., bug reports), and it can be collected periodically without losing quality.

- *Product metrics*: such metrics describe characteristics of the source code investigating a number of quality aspects such as its maintainability, its architecture, its complexity, etc. Basically, product metrics provide an analysis (at different levels of detail) of the structure of the source code and its evolution over time highlighting areas that are crucial for the overall system and the ones that need to be improved [24–27]. Such analyses are useful to perform several activities including:
  - *Assess the quality of the system*: data are collected at different levels of granularity providing information about different quality aspects at different levels (e.g., method, class, file, package, etc.). This kind of information can be compared to the levels that are accepted by the literature and by the specific requirements of the system developed. This allows the identification of areas of the source code that are likely to generate problems in the future and prevent them. Moreover, such analysis can be used to train newcomers to become confident to the different parts of the code and help them in becoming productive as soon as possible.
  - *Provide an overview of the evolution of the system*: the data collection can be performed at different stages of the development. Ideally, it can be performed every day to provide a continuous analysis of the evolution of the system and detect early possible problems that may arise. The analysis of the evolution of the system helps in the identification of architectural problems and helps in the definition of possible solutions.

Product metrics are extracted by different tools that are intended to collect different kinds of data. In any case, all the data are collected off-line connecting to the version control systems that store the source code. In many cases, the extraction of the data is performed through a static analysis of the source code. It is a good practice to store code that actually compiles and link properly to have all the data extracted since some tools require at least code able to compile correctly. However, many tools are able to extract data even from code that does not compile.

Besides the data collection, another important aspect is the integration of the collected information and the generation of visualizations that the different stakeholders find useful [28]. Different development teams and different companies have very different needs that require an extremely flexible reporting approach that may include the visualization through a web page, the generation of a custom report, the generation of a periodic email, etc. The flexibility of the reporting is important to allow the team to respond quickly to internal ever changing needs and to the inquiries coming from the management that prefers the traditional way of reporting about the status of a software project.

12.5 iAgile Benefits

12.5.1 Costs

Not all the cost reduction factors have been fully analyzed since many of them are related to human and social factors such as the growing capability of the production teams to deal with the specific product complexity. Relevant savings have been identified due to the implementation of a more effective production structure (resources and time) and a better understanding of the user needs/requirements [29].

12.5.2 Customer Satisfaction

The improved agile methodology described in this chapter is heavily based on a strong involvement of the user in the production cycle. The involvement is not limited to the initial phase where the user needs are defined and the user stories negotiated, but is continuously applied through the life cycle. Experts from the user communities are trained on the basic feature of iAgile and fully included in the production teams. During the sprint reviews, the "customer" top-level representative who is asked to accept the product delivery may receive the final presentation briefing by a component of his own community who is acting as team product owner.

12.5.3 Quality

The quality benefits introduced by iAgile are subtler to characterize than initially expected. In most agile methods, there is an initial assumption that quality will be higher because new code will be developed only when needed, immediately integrated, and eventually known by the majority of the development team. This assumption is not necessarily valid in iAgile, and in fact we argue that when multiple teams work on different code bases, the beneficial effects of such practices like pair programming and continuous integration are reduced because the physical separation of the teams creates a knowledge separation as well, whereas the code is inherently connected and often linked by unforeseen side effects that creep beyond the separation by protocols.

Thus, the error rate in our development is not expected to be lower than the one we measured in previous developments. However, we expect a marked improvement in the quality as perceived by the customer, because of the following fundamental reasons:

- The customer is able to witness development as it happens and is able to influence the development of his or her requirements in a direct and productive manner.
- The code base being developed using iAgile is smaller than the corresponding code base developed using a traditional method because only the requirements needed by the customer survive in the code base. Thus, the scope of the code base is more precise and responsive to the user needs.
- The faster feedback mechanism allows the user to signal errors early on in the development of a functionality, which increases the probability of passing verification and validation earlier than with other development methods.

12.6 Applying iAgile: A Case Study

To put the method described above to the test, we have decided to apply our method to a real defense project, thus creating a case study. This case study contains our experience in creating a command and control system for the 4th Logistic Division of the Italian Army's General Staff. This project was the first agile pilot the Army approved and was instrumental to investigate development cost reductions. Additionally, the constantly changing conditions in the theater of operations demanded an approach that could deal with rapid changes in user requirements, often mandated by previously unknown operational conditions. This experience has been continuing for about two and a half years, and although a first operational release has been issued, it is still being improved and expanded as the users gain more confidence and understanding of the possibilities of this means of software development.

12.6.1 Introduction to LC2EVO

In 2014, the Italian Army General Staff Logistic Department started the development of the Land Command and Control Evolution (LC2EVO) system. This evolutionary military command and control (C2) software system was based on the "Mission Thread"-based approach adopted by NATO.

This effort was generated by the urgent need to support the evolution of the land component of C2 achieving high "customer" satisfaction in a situation of extreme volatility of the user requirement. At the same time, it was necessary to substantially reduce the budget necessary for this software development and maintenance.

The Army software developers started the development using a commercial agile software development methodology available at the time: scrum. This methodology was, and still is, very successful in commercial environments, where it is often indicated as the method of choice for the majority of the Android- and Linux-based software application producers.

When we initially started, we had only one experimental scrum team building LC2EVO,, and we adopted production cycles of 3 weeks. The major key actors with programmers were subject matter experts, security specialists, a scrum master, and a product owner. The team components were taken from industry and military and were located at the army staff main facility. The production was extremely successful, and even the very first sprint, which was supposed to be only a start-up trial one, actually delivered the planned product.

Subsequently, though, the increasing product complexity and the growth of stakeholders' expectations made it necessary to create an ad hoc methodology. It had become very clear that the commercial scrum methodology was not capable of handling the peculiarity of a high-reliability software production with such an articulated and extended user community as the one in charge of the land C2 operational requirement.

A community of interests stemming from the Army but including experts from the universities and the defense industry conceived a customized agile software development process, iAgile, and applied it to the LC2EVO production. This methodology is currently shared with an even broader community including defense industries, universities, and the software engineers taking part in the Defense & Security Software Engineers Association (DSSEA).

The structure of the LC2EVO software is characterized by core services and functional area services (FAS). The core services are web based, and the FAS are derived by the Mission Threads defined in the International Security Assistance Force's Concept of Operations (ISAF CONOPs) and currently adopted by NATO. Core services and the individual FAS software components can be separately changed to accommodate the particular mission need defined by the military user. At the same time, all the FAS can share the data and the artifacts developed individually maximizing code reuse.

The first practical test for LC2EVO was in a NATO exercise at the Coalition Warrior Interoperability eXploration (CWIX 2015[1]), with very positive results, particularly in the area of software security.

In LC2EVO, the core service software segment provides integrated management of all operationally relevant events using a common cartography layer. Interoperability with all the necessary NATO-listed software packages is guaranteed. Collaborative generation of all the documents necessary for the command post activity is obtained largely by making use of common off-the-shelf (COTS) product. Garrison (static situation) infrastructural data flow for common daily homeland-based operations is fully integrated. Both the georeferenced military cartography and the web-based commercial ones are developed and integrate all the available information (weather, traffic, etc.).

The FAS are under continuous development and change for every release. The principal ones are as follows:

[1] www.act.nato.int/cwix

- Battle Space Management: designed to provide the C2 support for homeland security operations, as an embedded capability of tracking friendly secure mobile units using multiple types of cartography. Voice and messaging capability are implemented as well as an integrated NATO JOCWatch feature.

  The LC2EVO infrastructure FAS have been realized with large reuse of the code and functions realized before. The core capability of FAS is to provide an extensive and detailed set of functionalities needed to manage the army real estate.
- Joint Intelligence, Surveillance, and Reconnaissance (ISR): provides management and analysis functions for the intelligence preparation of the battlefield.
- Joint Fire and Targeting: supports all the coordination and planning activities related to the fire power support including all available effectors.
- Military Engineer and Countering of Improvised Explosive Device (IED): initially designed to support the collection and management of data about unexploded ordnance (UnExO) from WW2 found on the national territory, this was soon expanded to provide support to the counter-IED operations (attack the net and force protection).

This LC2EVO software segment has more than 1000 registered users from more than 600 military facilities in the homeland and abroad. Reuse is going to happen as the realization of the strategic tool Infologistica FAS, to be used to give to top-level military commander the real-time situation of all army materials, is completed.

12.6.2 Implementing iAgile

Although the consolidation of all procedures, tools, and methodological peculiarities is still ongoing, some elements have been clearly identified. Some of these elements will be now briefly discussed.

Although the project was initially started using a custom version of scrum, a number of relevant deviations from this methodology were almost immediately applied. One of the most significant ones concerns the user story collection process. It appeared that a simple set of interviews with selected users was not giving to the software engineers the expected quantity of information to decide which software functionalities were more relevant to the users. At the time, the identified issues were as follows: the very relevant expertise of users in their domains, the limitation of the natural language, the link with standard operating procedures, the incomplete understanding of the user of his own requirements, and of course the complexity of the command and control domain.

The high level of expertise in the user, although desirable from the point of view of requirement correctness, has two major shortcomings: (1) the user gives most of the domain knowledge for granted and omits many of the detailed descriptions (e.g., the user says “operation” to indicate the event occurring after the planning, but the specific kind of operation may vary from a convoy movement to an airborne

deployment of a special unit) and (2) the user knowledge is sub-domain specific and mostly experience based, which makes every user different from all the others in the same domain (e.g., a user with relevant experience in out-of-area operations will give more emphasis to the solution of the problems he or she encountered in the last mission, which may be outdated or too "theater specific").

The use of natural language is a point of strength in the agile methodology, but for the description of the user needs in complex domains, it may result in poor or meaningless user stories or in complicated epics with many links to external doctrinal documents which are difficult to understand for the software engineers. One of the methods implemented to correct the above situation was the definition of an iAgile empirical method for the product backlog definition. This method has not been completely formalized yet, but most of its steps emerged during the setup of the LC2EVO production structure and correlated procedures. The first component of the empirical methods is the user story quality evaluation [30]. The second component is the evaluation of the user characteristics: position in the organization (pure user-user/stakeholder-stakeholder). The empirical method works as follows, where a typical multiuser-structured domain is assumed where multiple story tellers contribute to the same product backlog (PB):

Given St_n the story teller n,
WSt_n the weight factor due to the hierarchical position in the User Organization
(a top hierarchical position generates higher priority for the specific User Story),
ISt_n the communication effectiveness of the Story Teller based on its capability to transfer the story teller knowledge to the team (measured in software tasks generated from the specific user story),
UsV User Story Value.

For a specific user story an empirical value can be estimated as:
$UsV_k = WSt_n * ISt_n$

For non-functional requirements (Nfr), a weight factor is then generated by the technical members to recognize the priority of each user story:
Nfr_k {statement}$_k$, where $0.1 < Nfr_k < 1$ (1 corresponding to obligation to execute Nfr associated tasks)

The initial product backlog PB resulting by the first round of interviews is then assembled to generate a list of N scrum-like statements, and will be a weighted list:
PBin= [UsV_1 {statement}$_1$... Nfr_k {statement}$_k$.... UsV_k {statement}$_k$, UsV_N {statement}$_N$]

This initial ranking will be used to generate the first prioritized PB, which shall be negotiated between the development team and the users/stakeholders' project board. This negotiation will take into account the initial values, but quite often the priority will be changed. To trace the evolution of the user stories due to nonlinear elaboration of the stories, a user story risk factor is generated (Sr_k) associated to the de-ranking of the story. So the first workable PB takes the following form:

$$PB1 = [UsV_1\ \{statement,\ Sr_1\}_1 \ldots Nfr_k\ \{statement\}_k \ldots UsV_k\ \{statement,\ Sr_k\}_k, \ldots UsV_N\ \{statement\ Sr_N\}_N]$$

At every production cycle (Sprint) the list is re-elaborated and the risk re-assessed.

12.6.3 Results and Discussion

On all accounts, the implementation of LC2EVO using iAgile was a success. This was ascertained both quantitatively and qualitatively; and although for comparison's sake the quantitative data are the most usable, sometimes the decision to continue with a method or practice is based preponderantly on qualitative decisions.

12.6.3.1 Costs

The foremost parameter that characterizes the need to adopt iAgile in the Italian Army is cost. The development of a command and control system in defense domains is evaluated at around US$ 85 per ELOC [31]. This figure is based on systems developed in third-generation languages, using conventional plan-based development processes. In the Italian Armed Forces, this figure is accepted, and in fact some of the previous projects turned out to be even more expensive. In particular, the Italian Air Force's command and control telecommunication system (CATRIN) cost US$ 65 per ELOC in 1992, which is about US$ 111 per ELOC when adjusted for inflation. A more recent project, the Italian Automated Information System for Command & Control, was completed in 2012 and cost a similar amount: US$ 100 per ELOC. In contrast, the development cost for LC2EVO was US$ 10 per ELOC.

It should be noted that LC2EVO was implemented in a modern language, whereas previous systems had been implemented in an imperative structured language, which should account for part of the lower development costs. However, we also have some additional hypotheses.

One of the reasons for this cost reduction may be attributed to the more effective production structure adopted by iAgile. Traditionally, a large number of activities in the production of a software system are not coding activities. For example,

conflict resolution, resource allocation, documentation production, customer interaction, and demonstrations often become so much part of the development process to take up to 40% of the total personnel time. For LC2EVO, this number is less than 10%. This means that more of the project cost is directly tied to the value perceived by the customer, thus making it possible to reduce activities that may seem important at first blush but reveal themselves to be less relevant in postmortem analyses.

Another reason for the cost difference may be due to the increased interaction with the user, which creates a bond through which user needs are better understood by the team and indeed by the user himself or herself, as the evolution of the product and its continuous testing by the user establishes a virtuous feedback circle. This translates in a very high percentage of requirements implemented eliciting customer approval: approximately 90% for LC2EVO, as compared to about 40% for conventional development methods [31].

Finally, it is possible that the short development cycles created a risk-limiting effect: requirement errors are caught early and corrected when they still have little impact on the project, because no requirement chaining has occurred yet, and thus the amount of rework needed to rewrite the requirement is limited to that requirement only instead of all the subsequent dependent requirements (Fig. 12.2). In fact, the amount of software produced in LC2EVO while a software requirement discrepancy exists is about one-fourth the amount of legacy methods. It is important to note that this particular effect is not a prerogative of iAgile, but is present in all agile and many iterative methods.

12.6.3.2 Customer Satisfaction/Quality

As hypothesized, the quality in terms of defect density or error rate did not differ significantly from that of the previous systems. What changed nearly radically was the acceptance and satisfaction by the customer. This change is to be expected when adopting a user-centric view and an iterative development model, as is the case in agile software development. We do not believe that iAgile differs from other agile methods in this respect.

However, the results prove that iAgile has in fact been able to extend the results of common agile methods to a domain where these methods were not used before. LC2EVO is a mission-critical software product with strong security constraints, and to our knowledge agile methods are not adopted for this kind of software. We believe that iAgile was successful in developing a successful mission-critical command and control product because of two different validations. Firstly, the requirements were validated with the user on a constant basis, resulting in a burndown chart that is similar to that of many successful agile product developments (Fig. 12.3).

Secondly, as previously mentioned, LC2EVO has been tested in the field by means of inter-force simulations (CWIX 2015 NATO exercise) and has behaved

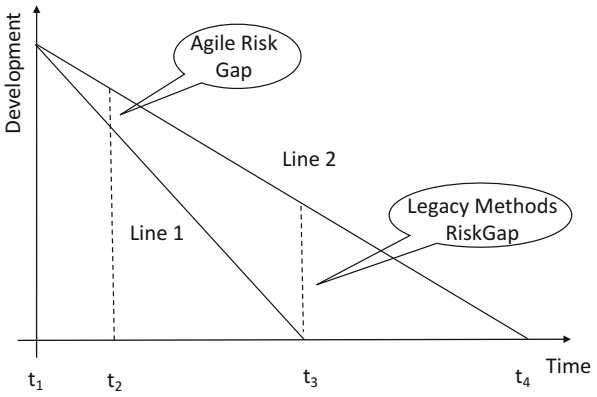


Fig. 12.2 Risk gap between iAgile and traditional methods

exactly as specified both in terms of functional requirements and of nonfunctional requirements, reliability in particular.

12.6.3.3 Learning Curve

Because it was the first time, we applied LC2EVO iAgile to a software development project, we expected to see some sort of learning curve. Figure 12.4 shows what we found while monitoring one of the seven development groups. The two control lines come from literature. It appears that there is a learning curve as it is possible to discern an overall climbing trend in the curve, but it is also apparent that software development in LC2EVO occurred at different speeds, displaying some efficiency peaks of over six ELOCs per developer per hour. This is definitely something worth checking in the future as it might be a function of the difference requirement complexity between sprints, but it could also be a dynamic specifically linked to iAgile.

12.7 Conclusions

This chapter describes iAgile, a software development and project management method that employs agile principles to make it possible to build highly secure, mission-critical software at lower costs and with higher customer satisfaction. This method is a unique combination of agile principles, innovative tools, structured user community, and a custom agile development doctrine that provides a change management path to adopt the process in a traditionally conservative environment while maximizing returns for the users and the engagement for developers.

iAgile is based on a proven enterprise development method, scrum; but the requirements of working in a mission-critical, highly specialized environment with a combination of consultants and army personnel required substantial

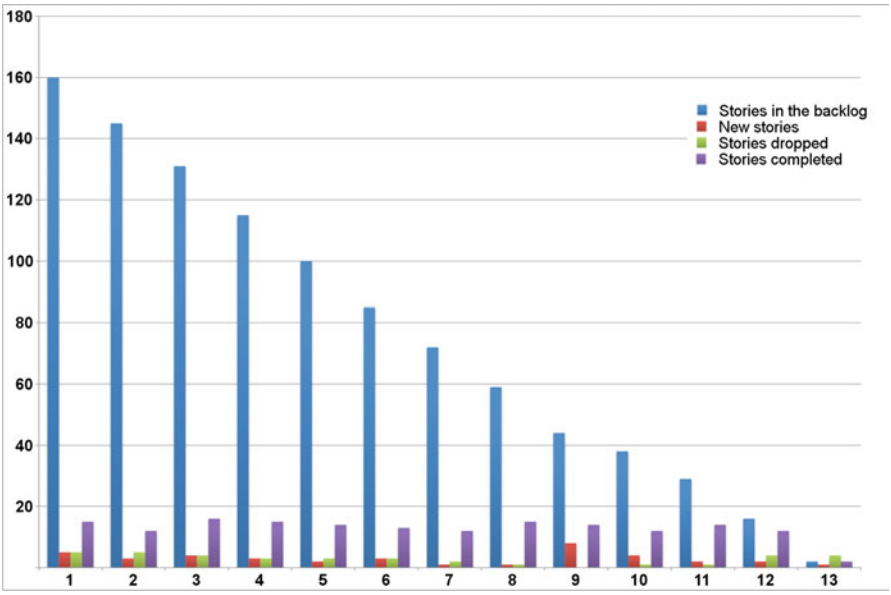


Fig. 12.3 Burndown chart for LC2EVO

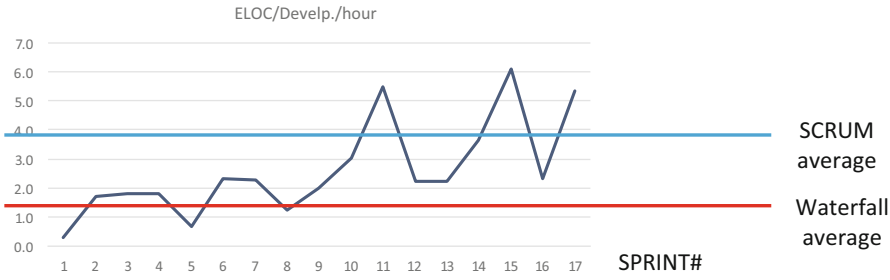


Fig. 12.4 Learning curve in LC2EVO development

changes. These include the introduction of a noninvasive measurement system to create an objective representation of the project and its progress, the specialization of the product owner into a product owner board and team product owners, the enhancement of the scrum master role, a screening method for the composition of each team, the creation of a social network of stakeholders to provide more varied opinions in complex decisions, the recording of scrum rituals, the adoption of change management practices to facilitate the introduction of iAgile in an armed forces context, specialized training, and the creation of a governance structure based on the user community.

The differences between scrum and iAgile necessitated the creation of a unique support infrastructure. The first pillar of this support infrastructure is the specialized training developed in collaboration with academia. The second pillar is the creation

of a series of innovative CASE tools, an effort which is still ongoing, and so far has produced a noninvasive measurement system and a requirement prioritization system. The third pillar is the support for a structured user community governance model, which generated a social network of professionals for reviewing the system and its progress from different points of view such as security, quality, etc. The fourth pillar is a custom agile development doctrine that enables the transferability of iAgile to other armed forces domains and/or groups.

To test iAgile we have adopted it to build LC2EVO, a command and control system for the Italian Army that is able to work in an inter-force context and is fully integrated with NATO communication protocols.

The results show that iAgile has worked as expected in this case, with an overall cost reduction of up to ten times compared with previous efforts in the same domain. However, these results have been demonstrated in one project only, albeit a 2.5-year-long one. Future work will be needed to extend the generality of these results. In particular, it will be necessary to implement iAgile in a completely different project from LC2EVO, to ensure the transferability of the methodology regardless of the team involved. Therefore, it is expected that in the Italian Army, future versions of LC2EVO will continue to be developed with iAgile and that the method will be adopted by more departments to respond to the needs of modern operations in the armed forces.

References

1. Bungay S (2011) The art of action. Nicholas Brealy Publishing, Boston
2. Waldner J (1992) Principles of computer-integrated manufacturing. Wiley, Chichester. ISBN 0–471-93450-X
3. Bicheno J, Holweg M (2009) The lean toolbox. PICSIE, Buckingham. ISBN 978–0–9541244-5-8
4. Agile Manifesto (2001) at http://www.agilemanifesto.org. Retrieved 10 July 2016
5. Chin G (2004) Agile project management. AMACOM, New York
6. Sutherland J, Viktorov A, Blount J, Puntikov N (2007) Distributed scrum: Agile project management with outsourced development teams. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS '07). IEEE Computer Society, Washington, DC, USA, p 274a-. DOI=http://dx.doi.org/10.1109/HICSS.2007.180
7. Hoda R, Noble J, Marshall S (2008) Agile Project Management. In: Proceedings of the New Zealand Computer Science Research Student Conference, Christchurch, New Zealand
8. Chagas LF, de Carvalho DD, Lima AM, Reis CAL (2014) Systematic literature review on the characteristics of agile project management in the context of maturity models. In: Mitasiunas A et al SPICE 2014, CCIS 477, Springer, Cham, pp 177–189
9. Dyba T, Dingsøyr T, Brede MN (2014) Agile project management. In: Ruhe G, Wohlin C (eds) Software project management in a changing world. Springer, Berlin. doi:10.1007/978-3-642-55035-5_11
10. Crowder JA, Friess S (2015) Agile project management: managing for success. Springer, Cham. doi:10.1007/978-3-319-09018-4
11. de Kort W (2016) DevOps on the Microsoft Stack. Apress. doi:10.1007/978-1-4842-1446-6_5
12. AXELOS (2009) Managing successful projects with PRINCE2

13. Tuley F (2015) The PRINCE2® Foundation PDF training manual. Management Plaza, Tremelo
14. Kotter J, Rathgeber H (2013) Our Iceberg is melting: changing and succeeding under any conditions, Pan MacMillan. ISBN 1447257464
15. Benedicenti L, Ciancarini P, Cotugno F, Messina A, Pedrycz W, Sillitti A, Succi G (2016) Applying scrum to the army – a case study. In: 38th International Conference on Software Engineering (ICSE 2016), Austin, 14–22 May 2016
16. Sillitti A, Janes A, Succi G, Vernazza T (2003) Collecting, integrating and analyzing software metrics and personal software process data. In: EUROMICRO 2003, Belek-Antalya, Turkey, 1–6 September 2003
17. Moser R, Pedrycz W, Sillitti A, Succi G (2008) A model to identify refactoring effort during maintenance by mining source code repositories. In: 9th International Conference on Product Focused Software Process Improvement (PROFES 2008), Frascati (Rome), Italy, 23–25 June 2008
18. Sillitti A, Succi G, Vlasenko J (2012) Understanding the impact of pair programming on developers attention: a case study on a large industrial experimentation. In: 34th International Conference on Software Engineering (ICSE 2012), Zurich, 2–9 June 2012
19. Astromskis S, Janes A, Sillitti A, Succi G (2014) Continuous CMMI assessment using non-invasive measurement and process mining. Int J Softw Eng Knowl Eng World Sci 24 (9):1255–1272
20. Coman I, Sillitti A (2007) An empirical exploratory study on inferring developers' activities from low-level data. In: 19th international conference on Software Engineering and Knowledge Engineering (SEKE 2007), Boston, 9–11 July 2007
21. Di Bella E, Fronza I, Phaphoom N, Sillitti A, Succi G, Vlasenko J (2013) Pair programming and software defects – a large, industrial case study. Trans Softw Eng IEEE 39(7):930–953
22. Fronza I, Sillitti A, Succi G (2009) An interpretation of the results of the analysis of pair programming during novices integration in a team. In: 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, 15–-16 October 2009
23. Sillitti A, Succi G, Vlasenko J (2011) Toward a better understanding of tool usage. In: 33th international conference on Software Engineering (ICSE 2011), Honolulu, 21–28 May 2011
24. Scotto M, Sillitti A, Succi G, Vernazza T (2004) A relational approach to software metrics. In: 19th ACM Symposium on Applied Computing (SAC 2004), Nicosia, Cyprus, 14–17 March 2004
25. Scotto M, Sillitti A, Succi G, Vernazza T (2006) A non-invasive approach to product metrics collection. J Syst Arch 52(11):668–675
26. Moser R, Sillitti A, Abrahamsson P, Succi G (2006) Does refactoring improve reusability? In: 9th International Conference on Software Reuse (ICSR-9), Turin, 11–15 June 2006
27. Moser R, Abrahamsson P, Pedrycz W, Sillitti A, Succi G (2007) A case study on the impact of refactoring on quality and productivity in an agile team. In: 2nd IFIP Central and East European Conference on Software Engineering Techniques (CEE-SET 2007), Poznań, Poland, 10–12 October 2007
28. Janes A, Sillitti A, Succi G (2013) Effective dashboard design. Cutter IT J Cutter Consortium 26(1):17–24
29. Messina A, Fiore F (2016) The Italian Army C2 Evolution from the current SIACCON2 Land Command & Control system to the LC2EVO using "agile" software development methodology, 2016 International Conference on Military Communications and Information Systems (ICMCIS), Bruxelles, 23–24 May 2016. doi:10.1109/ICMCIS.2016.7496585
30. Messina A, Marsura R, Gazzerro S, Rizzo S (2015) Capturing user needs for agile software development. In: Proceedings of 4th international conference in Software Engineering for Defence Applications, Rome, pp 307–319, doi:10.1007/978-3-319-27896-4_26
31. Reifer D (2004) Industry software cost, quality and productivity benchmarks. Reifer Consultants

Chapter 13
Ontology Annotation for Software Engineering Project Management in Multisite Distributed Software Development Environments

Pornpit Wongthongtham, Udsanee Pakdeetrakulwong, and Syed Hassan Marzooq

13.1 Introduction

A long-standing problem in multisite software development environments concerns the ways to tackle the disadvantages associated with remote communication and the coordination of software engineering projects. Each organisation has its own conventions and understandings of software engineering, and the actions of remote project members are often based on their understanding of the semantics of the multisite software development (who talks to whom, who does what, who knows about what), the content (which bugs affect which code, how the code structure fits together, why a code was implemented in a particular way) and the interactions (who said what, who did what). Within such a complex dynamic environment, shared understanding of software development is difficult to achieve. Different teams might not be aware of the tasks being carried out by others, potentially leading to problems such as an overlapping of work being carried out by two groups or work not being performed accurately due to a misinterpretation of the task. Wrong tasks may be carried out due to ignorance about whom to contact in order to obtain the proper details. If everyone who is working on a certain project is located in the same area, then situational awareness is relatively straightforward; however, for distributed international development, the overheads in communications incurred when meeting to discuss problems, raise issues, make decisions and find answers are very high. Consequently, these problems cause developmental delays and software problems since outstanding issues are not resolved and issues cannot be discussed immediately or in time, over a distributed team environment. There are many more issues in multisite software development, and no single project can

P. Wongthongtham (✉) • U. Pakdeetrakulwong • S.H. Marzooq
Curtin University, GPO Box U1987, Perth, WA 6845, Australia
e-mail: p.wongthongtham@curtin.edu.au

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_13

address them all, nor should it try to; otherwise the work will be shallow. Hence, communication and coordination issues are focused on in this research.

To help address this need for communication and coordination among different sites, software engineering ontology (SE ontology) has been developed. In field trials, this has been found to be effective in clarifying misunderstandings regarding concepts, relationships and project information [1]. The SE ontology defines common shareable software engineering knowledge and typically provides software engineering concepts: what the concepts are, how they are related and why they are related. When this generic ontology is specialised to a particular project and populated with instances which reflect the project information, it provides this common understanding of project information to all the distributed members of a development team in a multisite development environment.

In this chapter, the SE ontology is deployed in multisite software development environment. The development of SE ontology annotation is presented in this chapter in order to facilitate remote communication in multisite software development environments. SE ontology annotation (SEOAnno) is the process of assigning software engineering domain concepts to the development process of software products. It aims to clarify any ambiguity in remote communication. In other words, the SE ontology provides domain knowledge via the annotation process. The main artefact, which is centrally located and critical in multisite software development, is the source code. Hence, in this stage, the SEOAnno focuses on annotating the source code. Java source code annotation is chosen for a proof-of-concept development. Once the source codes have been annotated, they can then be used for communication among remote team members, among software agents and between members and software agents.

The use of the SE ontology-based approach is illustrated through a case study and validated in multisite software development. A use case is set in bug resolution for software maintenance and evolution. It involves the process of identifying, understanding, fixing a bug and taking additional steps in order to avoid the recurrence of similar bugs in the future. These most complex yet common activities associated with multisite software development demand an awareness of highly diverse kinds of software artefacts and stakeholders (e.g. component developer, deployment engineer, QA analyst, project/product manager, clients, etc.) distributed throughout different sites of the same project.

This chapter is organised as follows. Background knowledge including choice of research approaches is given in the next section. Literatures and existing relevant research are reviewed in Sect. 13.3. System architecture is presented in Sect. 13.4 followed by SEOAnno components in Sect. 13.5. Results are discussed in Sect. 13.6. The use of the SE ontology-based approach is illustrated and evaluated through a case study in Sect. 13.7. The chapter is concluded, with a discussion on future work, in Sect. 13.8.

13.2 Background

In this section, background knowledge pertaining to multisite software development and SE ontology is presented.

13.2.1 Multisite Software Development

Due to the emergence of the Internet and the globalisation of software development, there has been a growing trend towards multisite distributed software development. Software engineers working on the same project do not necessarily have to be co-located. They can be distributed across cities, regions or countries. For example, the requirement specification and design are done in Australia, the development is done in China and India, and the testing is done in Russia. There are several terms used for this approach: global software development (GSD), distributed software development (DSD) or multisite software development (MSSD). Ågerfalk et al. [2] discussed the reasons why organisations consider adopting distributed development of software systems and application models which include utilising a larger labour pool, accessing a broader skill base, minimising production costs and reducing development duration as a result of around-the-clock work schedules. Conchúir et al. [3] also mentioned other advantages such as market proximity, local knowledge and accessibility and adaptability to various local opportunities. However, this type of long-distance collaborative work is not without problems. It can cause challenges such as communication difficulties, coordination barriers, language and cultural differences [4]. This may result in some tasks not being carried out properly due to the difficulty of communication and coordination among team members located in different geographical areas and may lead to software project delay and budget overrun. Several efforts are being made to try to overcome these issues.

Thissen et al. [5] proposed communication tools and collaboration processes that were used to facilitate team communication and interaction in multisite software development environments. Biehl et al. [6] proposed a collaboration supporting framework named IMPROMPTU. It enabled remote members to discuss software development tasks through shared displays. Salinger et al. [7] developed an eclipse plug-in to support collaborative programming activities between distributed parties.

13.2.2 Software Engineering Ontology

Since the emergence of the Semantic Web, ontologies have been widely used to provide semantics and support the retrieval information based on the intended meaning rather than simply matching the search terms [8]. Ontologies have now

been applied in various fields including software engineering throughout the different phrases of the software development life cycle. Ontologies can provide a shared conceptualization of fundamental software engineering knowledge in the form of concepts, relationships and constraints. In addition, ontologies also have a great potential for analysis and design of complex object-oriented software systems by using them to create an object model for object-oriented software engineering [9].

In multisite software development environments, ontologies have played an important role in supporting collaborative work. There are several tools, techniques, models and best practices that utilise ontologies to facilitate collaboration, communication, project knowledge management and software engineering process activities. The resulting benefits, including effective communication among remote teams, knowledge sharing and effectiveness of information management, have been encouraging [10].

The SE ontology represents knowledge by structuring concepts, their relationships and their constraints; that is, it is an abstract level of representation. Project data over a period of time will be populated as instances in the deployment stage. More specifically, the SE ontology captures the generic software engineering concepts as well as the specific software engineering project information/data/agreements. The SE ontology contains a vocabulary of basic software engineering terms and a precise specification of what those terms mean. The SE ontology is populated with specific instances for a particular project for the corresponding software engineering concepts. These instances contain the actual project data being queried in the knowledge-based applications. Thus, the SE ontology includes the set of actual project data (i.e. instances of the concepts) and assertions that the instances are related to each other according to the specific relations between the concepts. This project will make use of the built SE ontology as well as refine the SE ontology.

The software engineering ontology comprises two sub-ontologies: the generic ontology and the application specific ontology [1, 11]. The generic ontology contains concepts and relationships annotating the whole set of software engineering concepts which are captured as domain knowledge. Application-specific ontology defines some concepts and relationships of software engineering for the particular software development project captured as subdomain knowledge. In addition, in each project, project information including project data, project understanding and project agreement that is specifically for a particular project need are defined as instance knowledge. Remote software teams can access software engineering knowledge shared in the ontology and query the semantic linked project information to facilitate common understanding and consistent communication.

13.3 Literature Review

13.3.1 Software Engineering Body of Knowledge vs Software Engineering Ontology

In relation to open knowledge on software engineering, there are two resources available, namely, (1) Software Engineering Body of Knowledge (SWEBOK) and (2) software engineering ontology (SE ontology). The SWEBOK [12] is a glossary of terms. It does not define the concepts or the relationships between the terms. In comparison, the SE ontology is organised by concepts, not words. This is in order to recognise and avoid potential logical ambiguities. It defines the concepts and interlinking relationships between each concept and the organisation of the whole body of knowledge, rather than a definition of each discrete term. The SE ontology is a big step forward. Nevertheless, both the SE ontology and the SWEBOK will provide benefits for a team of people working together, particularly if a person wants to find particular terms and has a good querying facility with which to do so.

13.3.2 Global Software Development

There are several works on global software development. Previous studies [13–15] have shown attempts to facilitate remote communications during software development. However, the communications are insufficient and ineffective due to a lack of consistent semantics across remote groups and a lack of shared knowledge and understandings. IBM created a social networking site called Beehive to help employees to share ideas, voice opinions and discover appropriately skilled people for a project – all of which are vital to working in distributed enterprises [16]. However, Beehive uses an informal and lightweight means of sharing knowledge. Hence, semantic-rich communications cannot be supported.

13.3.3 Technology-Supported Multisite Software Development

A multi-agent system has been used extensively to support collaborative software systems in multisite distributed software development environment. Various applications utilised multi-agent technology together with ontology to support software development activities. Col_Req was the multi-agent-based collaborative requirement tool that supported software engineers during the requirement engineering phase for collaborative acquisition, navigation and documentation activities [17]. Paydar and Kahani [18] proposed a multi-agent framework for automated testing of web-based applications. The framework was designed to facilitate the

automated execution of different types of tests and different information sources. Portillo-Rodriguez et al. [19] deployed agents and applied sociotechnical congruence measurement techniques to assist in resolving issues in a global software development. Lee and Wang [20] introduced an ontology-based computational intelligent multi-agent for Capability Maturity Model Integration (CMMI) assessment. It provided a summary of evaluation reports for the CMMI assessment. Nunes et al. [21] addressed the integration of a multi-agent system and software product lines (SPL) to support mass production of customised software. MADIS was introduced by [22] to support the distributed design process by managing information, integrating resources dispersed over a computer network and facilitating collaboration processes.

Recommendation system for software engineering (RSSE) has become an active research area. RSSE has been introduced to assist software developers to deal with huge amounts of information [23]. It can provide recommendations for development information (e.g. code, artefacts, quality measurement, tools, etc.) and collaborative information (e.g. people, awareness, status, priorities, etc.). Several works have focused on recommending experts/people. Ponzanelli [24] developed (i) a mechanism to collect data, (ii) a recommendation engine to analyse data and generate recommendations and (iii) a user interface to deliver recommendations. Xin [25] proposed a recommendation system to assist developers for bug resolution. Codebook [26] was a social network web service that linked developers and their work artefacts and maintained connections with other software team members. Conscius [27] was a recommender system that located a source code expert on a given software project by using communication history, source code, documentation and SCM change history.

Steinmacher et al. [28] proposed a system that can assist newcomers to discover the expert with the skill that matched selected particular issue in order to mentor the technical and social aspects of a particular task. Ensemble was an application that helped software team members to communicate in the current works by recommending other people when the developer updates any related artefacts such as source code or work items [29]. Other works focused on supporting developers while they were coding or debugging program. Fishtail was a plug-in tool for the Eclipse IDE which automatically recommended source code examples from the web to developers that were relevant to their current tasks [30].

Cordeiro et al. [31] proposed a context-based recommendation to support problem-solving in software development. They developed a client/server tool to integrate recommendation of question/answering web resources in the developer's work environment to provide automatic assistance when the exception errors occurred. DebugAdvisor [32] was proposed as a search tool for debugging which supported fat query, a query with all contextual information of the bug issue. Developers could do a bug report search from multiple software repositories with a single query. The system returns a bug description ranked list that matches the query and then uses it to retrieve recommendations for the related artefacts such as source code and functions from the generated relationship graph. Most RSSEs use traditional knowledge representation and syntactic matching approaches which can

cause the problem of ambiguity in keyword-based queries. The new generation of recommendation systems can benefit from the Semantic Web and ontologies to tackle the problem. Ontologies can be used for describing semantics of software engineering domain knowledge and project development information including software artefacts. Borges et al. [33] have studied along the same lines by utilising ontologies to support distributed software development enabling knowledge access and knowledge sharing among remote team members.

StakeSource [34] is a web-based tool that automates the StakeNet [35] approach for stakeholder analysis. It uses Web 2.0 technologies such as crowdsourcing and social networks to identify and prioritise stakeholders. Liu et al. [36] propose an automated approach for SOA design patterns by using an ontology knowledge base to provide a formal description of SOA design patterns. The system obtains user requirements in the form of questions and answers in order to avoid its complexity in natural language form. The recommendation is generated based on the user answers to the proposed questions and the sorting choice of property value in the constraint program.

Although there is substantial literature on multi-agent systems and ontology-based multi-agent systems for software engineering and the RSSE, however, the existing approaches focus on defining the context of the system, locating and retrieving the information, reasoning the knowledge and facilitating agents' communication and interoperability. We concentrate on assigning domain knowledge to the development process of software products aiming to clarify any ambiguity in remote communication. We link domain knowledge to software artefacts (i.e. code) in which it can then be used for project communication and coordination.

13.4 System Architecture

The conceptual architecture of the SEOAnno system as shown in Fig. 13.1 consists of three components – multisite development environment, SEOAnno and semantic repository – to store the semantically annotated structured data. Each of these components is discussed as follows.

13.4.1 Multisite Development Environment

A multisite development environment represents the integrated development environment (IDE) being deployed by a company (could be a software house or in-house IT department of an organisation) that supports the distributed multisite software development processes [37] and implements software configuration management (SCM) [38] to maintain the source codes (also known as configuration items in SCM). SCM is assumed to have version control to manage the code being accessed and maintain distributed code from multiple teams.

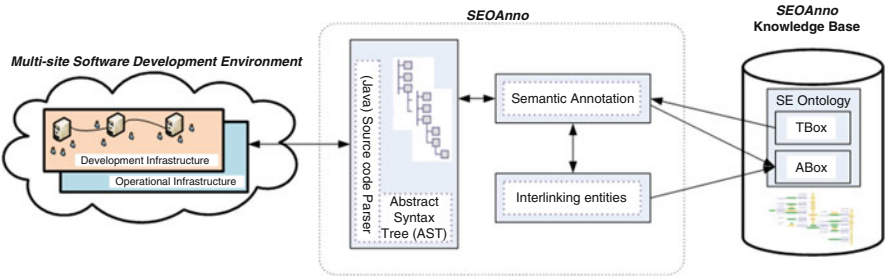


Fig. 13.1 Conceptual architecture of the SEOAnno system

13.4.2 Software Engineering Ontology Annotation (SEOAnno)

SEOAnno which is the core of the SEOAnno platform annotates the source code using metadata that is semantically rich to enable to interlink source codes with other relevant information. The other relevant information includes the identification of the key concepts being used in the source code and also referred to in the communication threads, mapping them with the international or de facto industry terminologies to establish a semantic relationship between different concepts. Essentially, SEOAnno has three functions to perform:

Identifying Key Concepts

- Identification of key concepts that are being used in the source code, source code-related documents or other software engineering artefacts that facilitates the software development activities
- Identification of key concepts that are being used or referred to in the communication thread
- Identification and selection of different vocabularies or metadata standards that can be used to further enrich the annotation of key concepts to provide a wider contextualization of the concept

Annotation of Key Concepts

- Semantically annotate the key concepts using software engineering ontologies to associate explicit semantic to the concepts that are defined and mentioned in the software artefacts.
- Enrich the annotation by describing the concepts using other popular ontologies and controlled vocabularies.

Interlinking Entities

- Interlink entities with similar entities defined in other datasets to provide an extended view of the entities represented by the concepts.
- Apply a range of semantic similarity constructs to connect related items [39].

13.4.3 Semantic Repository

A semantic repository represents the knowledge base which persists and updates the semantically rich annotated structured data. The information in the knowledge base can be classified into Tbox and Abox [40] to split different types of knowledge statements stored in the repository. Ontologies, which formalised the conceptualised knowledge of a particular domain, provide explicit semantics which are generally split concepts and their relationships from the instances and their attributes. Here, the Tbox represents terminological data that defines classes, properties (attributes), relationships (object properties) and axioms, whereas Abox represents assertion data that enumerates the individual instances of the concepts and defines them with factual statements. In this chapter and in general, the Tbox represents the schema and structural and intentional component of domain knowledge, and their separate treatment enables the application to support different services such as concept-based search, entailment to retrieve implied knowledge and consistency checking to avoid conflicting knowledge. The Abox represents the ontology instantiation where instance data is marked up with ontologies to help applications retrieve instance-related information to discover the attributes and class membership and relationship of different entities. We consider the SE ontology and other ontologies which are reused and included in the Tbox, and all the data that is annotated in these ontologies is considered as part of the Abox. Using the semantic repository, the application (clients) will be able to perform a full test search, query expansion, disambiguation of entities and retrieval of entity semantic descriptions.

13.5 SEOAnno Components

Most of the problems that are being faced within multisite software development projects such as the automatic retrieval of the software artefacts specific to a given context or identification of interlinked artefacts are addressable by making information machine understandable and processable. Based on the conceptual architecture presented, a framework is implemented to represent the knowledge that is in a machine-understandable structure and also interlinkable with other information sources. Figure 13.2 shows the different components and processes involved in the implementation of the SEOAnno solution to semantically annotate information related to software development for knowledge synthesisation, assimilation and

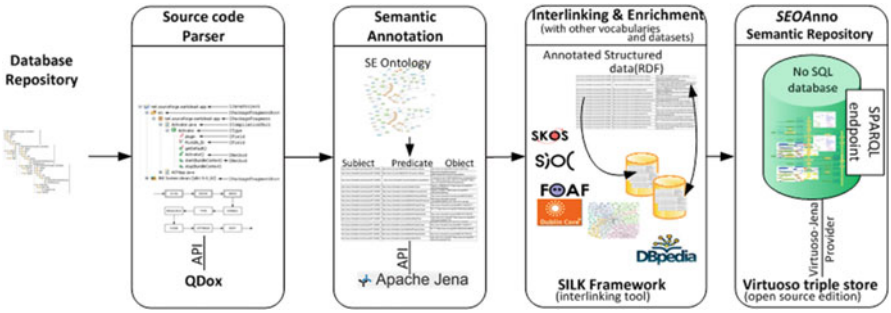


Fig. 13.2 SEOAnno solution components and information flow

dissemination. The SEOAnno solution is consists of source code parser, semantic annotation, interlinking and enrichment and semantic repository for knowledge management.

13.5.1 Source Code Parser

The main artefact that has the central and critical position in a multisite software development business or in software development in general is the source code. Source codes not only implement the business functionality but also embody the business logic intertwined in the programmed code. Parsing source code, aside from other nonstructured information sources such as documents, is essential to identify the different elements of the source code which follow the given programming language syntax. Theoretically speaking, there are two parsing stages, namely, the creation of a concrete syntax tree (CST) and of an abstract syntax tree (AST). The CST which takes the form of an ordered and rooted tree and represents the syntactic structure of a string according to some formal grammar is often considered too detailed for practical use; therefore, for source code, parsing the AST is considered. Parsers convert a CST into an abstract AST. The underlying (programming language) grammar constructs the tree to identify different elements of the source code. The different elements identified by the AST, which include classes, methods, fields, constructor, interface and in-line documentation, are then annotated based on their type. QDox which is an open source parser for Java source files is used to construct the AST, and its API enables the SEOAnno to identify the different types of elements in a given Java project. A Java project file is generally a folder containing different project-related artefacts including source file, packages, third-party APIs and configuration files. In this research, we focused on the source files and the documentation to annotate project artefacts and domain and business logic-related information.

13.5.2 Semantic Annotation

The identified issues faced by the multisite software development including the identification of software artefacts that are being discussed or interlinking with other relevant artefacts are addressed by describing them with SE ontology. Semantic annotation process assigns the metadata to the source code elements and other software engineering artefacts to make them machine processable and understandable. The process of semantic annotation initiates after the parser (source code parser) parses the source code to identify different constructs of the code. Upon their identification, semantic annotation annotates the elements with the appropriate concept defined in the SE ontology. While SE ontology is a comprehensive software engineering ontology and covers all the aspect of software engineering, the semantic annotation process also considers other relevant domain ontologies and controlled vocabularies to enrich their semantic description. The use of different relevant (even overlapping) ontologies helps in finding semantic similarities with other similar entities described in other semantic repositories (triple stores). The semantic annotation component makes use of Jena Semantic Web framework (now part of the Apache Software Foundation) to model the RDF (Resource Description Framework) graph and adds semantic descriptions using the SE ontology components. The source code elements extracted from the AST (constructed by QDox) are given to the Jena API to construct the RDF statements (in the form of <subject> <predicate> <object> or <subject> <predicate> "literal value") to semantically describe resources (here, resources mean the elements of the source code such as class, method, parameter, return type, etc.).

13.5.3 Interlinking and Enrichment

The adoption of domain ontologies and controlled vocabularies brings a reusability factor [41] of the knowledge to the fore which is one of the core contributions of ontology use. Considering that ontology reuse and interlinking with other relevant entities encourage information interoperability, therefore, where possible, reused ontologies can be adopted and used in the community to produce network effects. This was also highlighted in [42] that "ontologies exhibit positive network effects, such that their perceived utility increases with the number of people who commit to them which comes with wider usage". Considering the best practices proposed in the literature and using the Ontology Usage Analysis Framework (OUSAF) [41] which empirically analyses the use of ontologies and ranks them based on their usage, we reuse the ontologies, wherever possible. Within the interlinking and enrichment process, different vocabularies such as Friend of a Friend (FOAF) [43], Dublin Core (DC) [44], Simple Knowledge Organization System (SKOS) [45] and Semantically Interlinked Online Communities (SIOC) [46] are used to enrich the semantic description of resources annotated by the semantic annotation

component. In addition to ontology and vocabulary reuse, interlinking includes the semantic relationship between similar entities stored in other datasets.

13.5.4 SEOAnno Semantic Repository

The ultimate objective of SEOAnno is to semantically describe the software engineering and software development-related information for automatic knowledge acquisition and management. The aforementioned components retrieve the artefacts related to the software development project, parse the source codes, identify the source code elements and annotate the elements with the concepts and relationship defined in SE ontology and other ontologies. The components and processes construct the RDF graph that is stored in the semantic repository for persistence. For storage, the Virtuoso (open source edition) triple store is used to store the RDF triples, ontologies and schemas and expose them using a SPARQL endpoint. SPARQL endpoint enables other applications and users to access the knowledge base by posing SPARQL queries.

13.6 Results and Discussion

This section presents the output and discusses the test results.

13.6.1 SEOAnno Results

SEOAnno is capable of annotating any Java source code. It produces instances of the software engineering concepts captured in the SE ontology. Example of a Java source code annotation is illustrated in Fig. 13.3. In annotation process, SEOAnno creates instances of those concepts captured in the SE ontology. For example, Bicycle is an instance of class Class in the SE ontology; gear is an instance of class Field in the SE ontology; setCadence is an instance of class Method in the SE ontology; and so on. Likewise, there are others that can be annotated and created as instances of concepts. SEOAnno is the glue that binds SE ontology to the source code via metadata. In other words, SEOAnno attaches semantic metadata to the source code pointing to SE ontology concepts and properties.

Ontologies are structured as a graph, not only just a hierarchy tree, so that each concept (as instances and classes) has relationships between them through a hierarchy relationship, object properties and data-type properties. For example, relationship hasMethod relates the SE ontology class Class with the class Method, and relationship hasField relates the SE ontology class Class with the class Field as shown in Fig. 13.3. Class relationship also inherits its relationship to instance level;

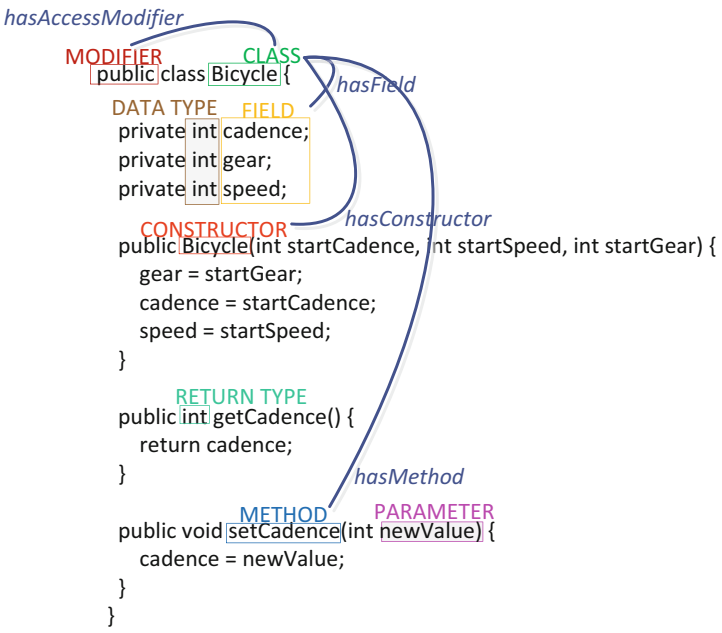


Fig. 13.3 A Java source code being annotated with concepts and their relationships captured in the SE ontology

hence, relationship hasField relates instance Bicycle (instance of class Class) with instances cadence, gear, and speed (instances of class Field). Likewise, relationship hasMethod relates instance Bicycle (instance of class Class) with instance setCadence (instance of class Method). Hence, as shown in Fig. 13.3, class Bicycle has seven relationships, i.e. relationship hasConstructor (Bicycle), three relationships of hasField (cadence, gear and speed), two relationships of hasMethod (getCadence and setCadence) and hasAccessModifier (public) relationship.

We annotate The HelloWorldSwing source code[1] which is simple and relatively short (Table 13.1). The output of the annotation in RDF format and in graph format for depiction is shown in Appendix A and Appendix B, respectively, and also in the SEOAnno website[2]. Another source code is Bicycle.java[3] which has complex and lengthy concepts embedded in the code. The output of the annotation in RDF format and in graph format for depiction is shown in Appendix C and Appendix D, respectively, and also in the SEOAnno website[2]. The full annotation output of both HelloWorldSwing and Bicycle code is available in the SEOAnno website[2].

[1] http://docs.oracle.com/javase/tutorial/uiswing/examples/start/HelloWorldSwingProject/src/start/HelloWorldSwing.java (visited Sept 2, 2016)

[2] http://seontology.wix.com/seoanno

[3] https://docs.oracle.com/javase/tutorial/java/javaOO/classes.html (visited Sept 2, 2016)

Table 13.1 Sample of annotation output

Source code	Annotation output
HelloWorldSwing.java	Method name: createAndShowGUI
...	
private static void createAndShowGUI() {	Method return type: void
	void is primitive type
...	createAndShowGUIMethod is STATIC
	createAndShowGUI modifier: static
	createAndShowGUI modifier: private
Bycicle.java	Method name: Bicycle
...	
public Bicycle(int startCadence, int startSpeed, int startGear) {	Bicycle modifier: public
	Parameter name: startCadence
	Parameter type: int
gear = startGear;	Parameter (is array?): false
cadence = startCadence;	Parameter name: startSpeed
speed = startSpeed;}	Parameter type: int
...	Parameter (is array?): false
	Parameter name: startGear
	Parameter type: int
	Parameter (is array?): false

13.6.2 Discussion

We implement a prototype of the SEOAnno platform and test it on the two recognised Java programs distinguished by the size of the source code. The simple and short Java code of HelloWorldSwing makes it easy to check the competence. The complex and rather long Java code of Bicycle is used to demonstrate its scalability. The SEOAnno covers 10 Java constructs, 34 relationships and 20 instance data in HelloWorldSwing Java code, while it covers 25 Java constructs, 168 relationships and 102 instance data in Bicycle Java code. In the next section, we present the use of the SE ontology-based approach in multisite environment.

13.7 Case Study

We illustrate our SE ontology-based approach through a case study in a multisite software development environment in which the scenario is depicted in Table 13.2.

A multinational software company was running a multisite software development project (a vehicle registration) at four sites: Perth, Shanghai, Dublin and

Table 13.2 A case study scenario

Date	Actor	Site	Action
25-Jul-15			Release of V1.1 Build 20150725
3-Aug-15	Richard	@Perth	Found a bug
			Filed the bug (# 873) with high severity
4-Aug-15	Richard	@Perth	Filed another urgent request (# 880)
4-Aug-15	Vishay	@Bangalore	Bug report opened
			Quick fixed and added a comment at the end of thccccce report
			Put report into the status of re-evaluation pending
11-Aug-15	Arleno	@Shanghai	Filed a duplicate bug (# 904)
13-Aug-15	Arleno	@Shanghai	Recognised as bug # 904 is a repeated report of bug # 880
			Realised that the solution (being a temporary workaround) was not good enough
15-Aug-15	Arleno	@Shanghai	Discussed the issue with his team members and supervisor, who added comments to report # 880 and directed their concerns back to the Bangalore Lab
17-Aug-15	Larry	@Bangalore	Provided another bug fix solution
22-Aug-15	Michael	@Dublin	Pointed out that Larry's fix might incur deadlocks in another related component and suggested reverting to the first fix
23-Aug-15	Larry	@Bangalore	Fixed the bug based on Michael's instruction
24-Aug-15	Michael	@Dublin	Checked the fix, marked the bug report status as "resolved" and closed the bug
24-Aug-15	Lisa	@Shanghai	Suggested that the latest fix resulted in a connection time out
25-Aug-15	Larry	@Bangalore	Asked for the effected component Lisa mentioned
25-Aug-15	Michael	@Dublin	Fixed the bug and explained his fix
29-Aug-15	Richard	@Perth	The bug was finally determined as resolved
		@Perth	The bug has led to a discussion in all four sites about the architecture of the effected component library
		@Bangalore	
		@Shanghai	
		@Dublin	

Bangalore. After releasing V1.1 Build 20150725, a bug was found by Richard@Perth. Richard immediately filed the bug in the project's issue tracking system. Given its high severity, Richard on the next day filed another urgent request in the issue tracking system, hoping to increase its priority ranking so that it could draw greater attention from developers. The bug report was soon opened on the same day by Vishay@Bangalore. He came up with a quick fix and added a comment at the end of the report, giving the report the status of "re-evaluation pending". One week later, Arleno@Shanghai filed a duplicate bug which was soon recognised as a repeated report 2 days later. Arleno then realised that the solution

(being a temporary workaround) was not good enough. He discussed the issue with his team members and supervisor, who added comments to report and directed their concerns back to the Bangalore Lab. Based on Arleno's detailed information, Larry@Bangalore soon provided another bug fix solution. This fix is then picked up by Michael@Dublin, a technical lead who used to work with the component in the software package. Michael pointed out that Larry's fix might produce deadlocks in another related component and suggested reverting to the first fix. The next day, Larry soon fixed the bug based on Michael's instruction. Michael checked the fix and marked the bug report status as "resolved" and closes the bug. The same day, Lisa@Shanghai stated that the latest fix resulted in a component connection time out. Later on, Larry asked her to explain on effected component. At this point, Michael stepped in, fixed the bug and explained his fix. Few days later, the bug was finally determined as "resolved" by Richard. The bug had led to a discussion in all four sites about the architecture of the component library.

In the above example, the bug was not a very complicated one and required only a straightforward solution. However, it took 26 days to fix this single bug and delayed the final product release. Many of the issues arising from the lack of common semantics discussed above have resulted in a similar lengthy process. First, the information relating to a bug was located in multiple social interactions, i.e. three discussions on the same topic (the bug) that have not been correlated with each other. This is also the reason that the same bug has been reported twice. Second, the bug was firstly found and solved by someone at some site without expertise in this specific area, which led to two iterations of invalid fixes before the expert, Michael, stepped in and finalised it. The failure to appropriately match expertise (people and sites) with problems in a multisite environment unnecessarily delayed the bug fix. Lastly, the lack of adequate knowledge sharing also played a role in delaying the fixing of the bug. Larry was unsure of what the affected component was, and there was no active knowledge support to explain what it does and how it might have affected the other component.

We then use this scenario as a case study to validate the framework. Distributed team worked on this multisite project having developers in Shanghai, Perth, Dublin and Bangalore. In order to demonstrate and validate the application of SEOAnno, the case study was based on a vehicle registration solution that was being developed by a distributed team working on this comprising of developers in Shanghai, Perth, Dublin and Bangalore. Figure 13.4 shows the class diagram of a vehicle registration application covering the main components of any normal Java-based project.

Vehicle registration is the main call which invokes either *Car* or *MotorBike* classes depending on what the user selects. Both these concrete classes implement *Information* interface and extend the *Vehicle* super class which provides generalised features of any type of vehicle. The project is developed in a multisite environment, and the location of each developer is depicted in Fig. 13.4, i.e. Vishay is working at the Bangalore site, Richard at the Perth site, Arleno at the Shanghai site and Alex at the Dublin site.

The SEOAnno solution is capable of semantically annotating the software development environment which includes project details and the project team

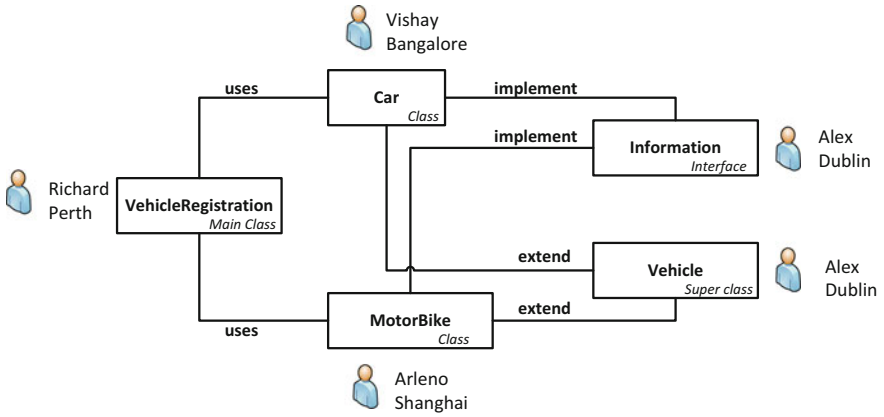


Fig. 13.4 Vehicle registration class diagram in multisite software development environment

including project management and development team, software configuration management and software engineering artefacts. For implementation and evaluation, vehicle registration project (as shown in Fig. 13.4) is used to semantically describe this project as well as the project communication. The following aspects of the project are represented and described using the SE ontology together with FOAF, DC and vCard ontologies/vocabularies as part of SEOAnno platform:

- Project description (project name, description, code and project management team)
- Project development team (developer name, location, time zone, role and list of source code on which she/he worked)
- Software artefacts (class, super class, interface, method, field, constructor)
- Software bugs database (bug/issue name, type and description)
- Communication focusing on bug and issues

The SEOAnno captures the above-mentioned aspects and builds a knowledge base to represent project- and communication-related information that is structurally described and semantically annotated. OpenLink Virtuoso (open source edition) is used by the SEOAnno to store in RDF graphs and the information related to the above-mentioned projects. Virtuoso's SPARQL endpoint is used to query the knowledge base and extract information. Different scenarios sufficiently represent the issues faced in multisite distributed software development environments discussed in this work. We demonstrate the issues through use case scenarios and address them by querying the knowledge base which provides the required and insight information.

13.7.1 Retrieve the Bugs Reported for a Class (Single Bug)

Before a bug is filed, a team member or a software agent could attempt to locate related problems from the issue tracking systems based on their associated class or component defined in the SE ontology and its instances. In this way, software problems are all related, and duplicated reports could be avoided, which will dramatically reduce confusion and unnecessary information overload. The duplication can be detected automatically by reasoning the result set.

Below is shown SPARQL querying to find any bug(s) related to class Car. In Fig. 13.5, the knowledge base is accessed to retrieve the bug(s) reported for a given software artefact. In this scenario, the query retrieves all the bugs found and reported that are related to the class Car. Figure 13.6 displays the record found in triple store (database) against the query previously mentioned. It found that the bug with id BugX001 has a major severity level, with a status "Open", and the key of the resource who has been assigned (or worked on) this bug.

The information demonstrates the ability of the system to provide all the information which a developer or any member of the software development team needs to know about the previously reported bug. This information will help to prevent reporting the same bugs multiple times, to retrieve information about previous bugs and to know who has fixed the bug. The availability of such information is essential for any multisite software development project.

13.7.2 List All the Bugs Reported for a Class (Multiple Bugs)

This scenario is similar to the above scenario although here we demonstrate the presence of multiple bugs for a given class, i.e. the main class. Figure 13.7 shows the SPARQL query used to retrieve the required information from the knowledge base. Figure 13.8 depicts the results of the query mentioned above. In this scenario, we can see that there are two (multiple) bugs reported for the main class. As mentioned in the above scenario, such information is important for multisite software development and helps to effectively resolve any issues related to a bug or communication.

13.7.3 Search for a Specific Type of Bug

In a large enterprise software development project, it is very important to know the different types of bugs that are being reported either by the software testing team or by end user. This helps to get to the root cause because it is also possible that the bugs are not related to the application itself but to the underlying infrastructure or network that is being used to run the application. Figure 13.9 shows a SPARQL

```
Select distinct ?name, ?pro, ?desc
where{
    ?cls rdf:type seo:CLASS.
    ?cls foaf:name ?name.
    ?cls seo:hasBug ?bug.
    ?bug ?pro  ?desc
FILTER regex(?name,'Car','i')}
```

Fig. 13.5 SPARQL query to display the reported bugs related to class Car

name	pro	desc
Car	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.seontology.org/activeseo/seo/v1#BUG
Car	http://xmlns.com/foaf/0.1/name	Car
Car	http://www.seontology.org/activeseo/seo/v1#associatedTo	http://www.seontology.org/activeseo/seoanno/data/Car
Car	http://www.seontology.org/activeseo/seo/v1#bugID	BugX001
Car	http://www.seontology.org/activeseo/seo/v1#bugDescription	This bug is related to the calculation of interest for Car insurance
Car	http://www.seontology.org/activeseo/seo/v1#hasBugSeverity	http://www.seontology.org/activeseo/seo/v1#Major
Car	http://www.seontology.org/activeseo/seo/v1#hasBugStatus	http://www.seontology.org/activeseo/seo/v1#Open
Car	http://www.seontology.org/activeseo/seo/v1#hasFixing	http://www.seontology.org/activeseo/seoanno/data/BugFixerResource
Car	http://www.seontology.org/activeseo/seo/v1#hasBugType	http://www.seontology.org/activeseo/seoanno/bug/type/LogicError

Fig. 13.6 Result of SPARQL query shown above

```
select distinct ?name, ?pro, ?desc
where{
  ?cls rdf:type seo:CLASS.
  ?cls foaf:name ?name.
  ?cls seo:hasBug ?bug.
  ?bug ?pro  ?desc
 FILTER regex(?name,'Main','i')
}
```

Fig. 13.7 SPARQL query to display the reported bugs related to main (program) class

name	pro	desc
MainClass	http://www.seontology.org/activeseo/seo/v1#hasBugType	http://www.seontology.org/activeseo/seoanno/bug/type/Standards
MainClass	http://www.seontology.org/activeseo/seo/v1#hasFixing	http://www.seontology.org/activeseo/seoanno/data/BugFixerResourceForBugX003
MainClass	http://www.seontology.org/activeseo/seo/v1#bugDescription	"This bug is related to the incorrect requirements. The implemented requirement is quite different from the one written in SRS. For any requirement change, approved CR is required."
MainClass	http://www.seontology.org/activeseo/seo/v1#hasBugStatus	http://www.seontology.org/activeseo/seo/v1#Open
MainClass	http://www.seontology.org/activeseo/seo/v1#hasBugStatus	http://www.seontology.org/activeseo/seo/v1#Cancelled
MainClass	http://www.seontology.org/activeseo/seo/v1#associatedTo	http://www.seontology.org/activeseo/seoanno/data/MainClass
MainClass	http://www.seontology.org/activeseo/seo/v1#hasBugType	http://www.seontology.org/activeseo/seoanno/bug/type/IncorrectRequirements
MainClass	http://www.seontology.org/activeseo/seo/v1#bugDescription	"This bug is related to the wrong computation of the interest. From computational point of view class calculated correct numbers but referring to policy document it seems to be not updated to new interest rates. On certain test data (using long digit number) it also through exception"

Fig. 13.8 Result of SPARQL query shown above

MainClass	http://www.seontology.org/activeseo/seo/v1#bugID	"BugX003"
MainClass	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.seontology.org/activeseo/seo/v1#BUG
MainClass	http://www.seontology.org/activeseo/seo/v1#bugID	"BugX004"
MainClass	http://www.seontology.org/activeseo/seo/v1#hasFixing	http://www.seontology.org/activeseo/seoanno/data/BugFixerResourceForBugX004
MainClass	http://www.seontology.org/activeseo/seo/v1#hasBugSeverity	http://www.seontology.org/activeseo/seo/v1#Major
MainClass	http://xmlns.com/foaf/0.1/name	"MainClass"

Fig. 13.8 (continued)

Fig. 13.9 SPARQL query to display bugs of a specific type

```
1  Select distinct ?name, ?bType, ?bTypeName,
2                  ?bTypeDesc, ?bClass, ?bDesc,
3                  ?bSev, ?bStatus
4  Where{
5     ?bug   rdf:type              seo:BUG.
6     ?bug   foaf:name             ?name.
7     ?bug   seo:hasBugType        ?bType.
8     ?bType seo:bugTypeName       ?bTypeName.
9     ?bType seo:bugTypeDesc       ?bTypeDesc.
10    ?bug   seo:associatedTo      ?bClass.
11    ?bug   seo:bugDescription    ?bDesc.
12    ?bug   seo:hasBugSeverity    ?bSev.
13    ?bug   seo:hasBugStatus      ?bStatus.
14    FILTER regex(?bType,'LogicError','i')
15 }
```

query which retrieves all the reported bugs of type "LogicalError". Query also accesses the bugs database to provide the textual description of a bug to help the user understand the nature of the bug that is useful for certain access items.

Figure 13.10 displays the instances of LogicalErrors in the database. It found that there are two classes in the project that have been reported to have logical error, and their severity and status have been also retrieved. The availability of such

information helps in understanding the overall quality of the project and planning further development accordingly. This will help to prevent the occurrence of similar bugs in software and to communicate with software developers working in distributed locations. A real-time bug statistic report can be developed and made available to stakeholders as a means of preventing such issues and improving overall project quality

13.7.4 Display the Components of a Given Software Artefact, i.e. Class

After a bug is filed, a team member or a software agent could look at links of related classes or components to see the possible impact of the fix on the class. Class relationships are all annotated in the SE ontology, so if a software agent is used, it would warn about the possible impact of the fix on the related software component based on the relationships captured in the SE ontology. Furthermore, by querying bug history (who added and updated it) and information about the component (e.g. subcomponents, relationships with other components), the relevant people check can be done before making any changes.

A full record of mappings between previously reported bugs and people/site that resolved those bugs can be kept to determine the mapping similarity score. In this way, it is able to recommend the persons who are most likely able to solve the bug problem. For example, when a bug was reported, it could have been brought to the attention of an expert instead of going through a circle involving a number of people who do not possess sufficient expertise to fix the bug. Moreover, it could embed the expert's details to the bug report so that before anyone attempted to fix it (since, due to company policy, only authorised developers can change certain parts of the code, so the expert might not be able to fix the bug himself), the expert could be directly consulted first.

Figure 13.11 shows a SPARQL query that displays all the attributes of a (Java) class. In this query, we retrieve the details which include author of class, version detail, access specifiers, number of methods it has, field members, constructors and other properties such as whether the class implements an interface and/or extends the superclass. It also tells whether any bug has been reported for this class.

Figure 13.12 shows the attributes and properties of Car class. Query result displays the author, i.e. Vishay who developed the class; the present version of class; access specifiers of class, i.e. public; the two methods this class has, i.e. getInformation and getMakeYear; and bugs reported for this class. As mentioned in the above scenarios, the availability of such insight is important to support activities associated with software development.

name	ProxySever	Car
bType	http://www.seontology.org/activeseo/seoanno/bug/type/LogicError	http://www.seontology.org/activeseo/seoanno/bug/type/LogicError
bType-Name	LogicError	LogicError
bTypeDesc	Missing or inadequate or irrelevant or ambiguous functionality in source code	Missing or inadequate or irrelevant or ambiguous functionality in source code
bClass	http://www.seontology.org/adtiveseo/seoanno/car	http://www.seontology.org/adtiveseo/seoanno/car
bDesc	This bug is related to the calculation of interest for car insurance	This bug is related to the calculation of interest for car insurance
bSev	http://www.seontology.org/activeseo/seoanno/seo/v1#Major	http://www.seontology.org/activeseo/seoanno/seo/v1#Major
bStatus	http://www.seontology.org/activeseo/seoanno/seo/v1#Open	http://www.seontology.org/activeseo/seoanno/seo/v1#Open

Fig. 13.10 Result of SPARQL query shown above

```
1  select distinct *
2  where{
3     ?cls   rdf:type    seo:CLASS.
4     ?cls   foaf:name   ?cName.
5     OPTIONAL{?cls seo:author              ?cAuthor.    }
6     OPTIONAL{?cls seo:version             ?cVersion.   }
7     OPTIONAL{?cls seo:hasAccessModifier   ?cAcessSpec. }
8     OPTIONAL{?cls seo:hasMethod            ?cMethod.    }
9     OPTIONAL{?cls seo:hasConstructor       ?cConst.     }
10    OPTIONAL{?cls seo:implementsInterface ?cImplInterface. }
11    OPTIONAL{?cls seo:hasParentClass       ?cHasParent.}
12    OPTIONAL{?cls seo:hasBug               ?cHasBug.    }
13    FILTER regex(?cName,'Car','i')
14 }
```

Fig. 13.11 SPARQL query to display all the attributes of a software artefact (Class properties)

13.7.5 Retrieve and Display Available Information of a Class

Since the information is stored in a structured format that is semantically describable, it can be easily interlinked with other relevant information to provide a real-time intelligent dashboard, a unified and consolidated snapshot of the project and

cls	http://www.seontology.org/activeseo/seoanno/data/Car	http://www.seontology.org/adtiveseo/seoanno/data/Car
cName	Car	Car
cAuthor	Vishay	Vishay
cVersion	1.2	1.2
cAccessSpec	http://www.seontology.org/activeseo/seoanno/seo/v1#Public	http://www.seontology.org/activeseo/seoanno/seo/v1#Public
cMethod	http://www.seontology.org/activeseo/seoanno/data/getInformation	http://www.seontology.org/activeseo/seoanno/data/getMakeYear
cConst		
cImplInterface	http://www.seontology.org/activeseo/seoanno/data/Vehicle	http://www.seontology.org/activeseo/seoanno/data/Vehicle
cHasParent		
cHasBug	http://www.seontology.org/activeseo/seoanno/data/BugIn_Car	http://www.seontology.org/activeseo/seoanno/data/BugIn_Car

Fig. 13.12 Result of SPARQL query shown above

tailored information. Figure 13.13 shows a SPARQL query to retrieve all the available or context-specific information of a given class. The query result displays class detail, developer detail and bug(s) detail location in which the code is developed.

Figure 13.14 displays context-relevant information of Car class. The query result displays the class name, access specifiers, version, location, time zone, customised document type used in documentation, interface and super class details, bugs-related information and the developers who work on the class as well as on bugs.

The availability of a rich set of information is essential for the development of a smart dashboard to provide the erudite insight about any multisite software development project. The above-mentioned five scenarios demonstrate the effectiveness and validity of the SEOAnno system which is considered as an essential component in providing an effective and efficient platform for distributed software development.

13.8 Conclusion and Future Work

The SEOAnno framework has been presented through the use of semantic representation of software artefacts and project source code. The framework facilitates remote communication and coordination in multisite software development

```
Select distinct ?name, ?pro, ?desc
Where{
  ?cls rdf:type seo:CLASS.
  ?cls foaf:name ?name.
  ?cls seo:hasBug ?bug.
 {
  ?cls    ?pro  ?desc.
 }
 UNION{
  ?bug seo:hasFixing ?fixer.
  ?fixer ?pro  ?desc.
  }
 UNION{
 ?bug seo:hasBugType ?bugType.
 ?bugType ?pro   ?desc.
 }
 FILTER regex(?name,'Car','i')
}
```

Fig. 13.13 SPARQL query to display the resource who has worked on resolving reported issues

environments. Source code was chosen in the annotation process due to its main artefact that is centrally located and critical in multisite software development. In future work, other artefacts such as diagrams can also be annotated given that SE ontology is used to provide domain knowledge. A proof-of-concept prototype implementation has been done in Java source code annotation. Types of source code other than Java can be included in future work. Experimental study of software engineers' productivity can be further investigated and is marked as future work. To measure the efficiency of SEOAnno platform, parameters include overall project completion time and decrease in issues, and issue resolution time can be considered. The metrics to measure project completion time include project size in man-days, line of code, software artefacts in number of documentations' pages, number of use cases and test cases and estimated and actual completion time in man-days. Effective communication reduces the time required to resolve issues reported by a remote team or users. The metrics demonstrate the efficiency in terms of project size in man-days, number of issues reported, average issue resolution time in hours (time between issue reporting and issue resolution), number of issues assigned to remote developers and number of issues closed without further escalation. The practical value and usability of the platform can also be evaluated for multisite software engineering. Surveys and focus groups can be used to determine any issues with the platform in order to probe in-depth issues. The platform can then be refined and shaped into a commercial grade system.

name	pro	desc
Car	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.seontology.org/activeseo/seo/v1#CLASS
Car	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://purl.org/dc/dcmitype/Software
Car	http://www.seontology.org/activeseo/seo/v1#hasAccessModifier	http://www.seontology.org/activeseo/seo/v1#Public
Car	http://www.seontology.org/activeseo/seo/v1#hasMethod	http://www.seontology.org/activeseo/seoanno/data/getInformation
Car	http://www.seontology.org/activeseo/seo/v1#hasMethod	http://www.seontology.org/activeseo/seoanno/data/getMakeYear
Car	http://xmlns.com/foaf/0.1/name	Car
Car	http://www.seontology.org/activeseo/seo/v1#inLineComment	This class is the class representing the registration process for Cars. This class extends Information class and implements Vehicle interface.
Car	http://www.seontology.org/activeseo/seo/v1#author	Vishay
Car	http://www.seontology.org/activeseo/seo/v1#version	1.2
Car	http://www.seontology.org/activeseo/seo/v1#location	Bangalore, India
Car	http://www.seontology.org/activeseo/seo/v1#timezone	GMT+5.30
Car	http://www.seontology.org/activeseo/seo/v1#doctags	@role developer
Car	http://www.seontology.org/activeseo/seo/v1#implementsInterface	http://www.seontology.org/activeseo/seoanno/data/Vehicle
Car	http://www.seontology.org/activeseo/seo/v1#hasBug	http://www.seontology.org/activeseo/seoanno/data/BugIn_Car
Car	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.seontology.org/activeseo/seo/v1#BUGFIXER

Fig. 13.14 Result of SPARQL query shown above

Car	http://www.seontology.org/activeseo/seo/v1#hasDeveloper	http://www.seontology.org/activeseo/seoanno/data/VRDev01Vishay
Car	http://www.seontology.org/activeseo/seo/v1#fixedDate	10Jan2014
Car	http://www.seontology.org/activeseo/seo/v1#fixedBy	Vishay
Car	http://www.seontology.org/activeseo/seo/v1#fixedBy	Developer-Name
Car	http://www.seontology.org/activeseo/seo/v1#fixDescription	These are the comments of resource hwo fixed it
Car	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.seontology.org/activeseo/seo/v1#BugType
Car	http://www.seontology.org/activeseo/seo/v1#bugTypeName	Logic Error
Car	http://www.seontology.org/activeseo/seo/v1#bugTypeDesc	Missing or Inadequate or irrelevant or ambiguous functionality in source code

Fig. 13.14 (continued)

References

1. Wongthongtham P et al (2008) Development of a software engineering ontology for multi-site software development. IEEE Trans Knowl Data Eng 21:1205–1217
2. Ågerfalk PJ et al (2005) A framework for considering opportunities and threats in distributed software development. In: Proceedings of the International workshop on distributed software development. Austrian Computer Society, Paris, 29 Aug 2005
3. Conchúir EÓ et al (2009) Global software development: where are the benefits? Commun ACM 52(8):127–131
4. Islam S, Joarder MMA, Houmb SH (2009) Goal and risk factors in offshore outsourced software development from vendor's viewpoint. Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on. Limerick, Ireland
5. Thissen MR et al (2007) Communication tools for distributed software development teams, Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce. ACM, St. Louis, pp 28–35
6. Biehl JT et al (2008) Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, Florence, pp 939–948
7. Salinger S et al (2010) Saros: an eclipse plug-in for distributed party programming, Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering. ACM, Cape Town, pp 48–55
8. Dillon TS, Chang E, Wongthongtham P (2008) Ontology-based software engineering- software engineering 2.0. In: Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on. Perth, Australia

9. Blanco-Fernández Y et al (2008) A flexible semantic inference methodology to reason about user preferences in knowledge-based recommender systems. Knowl-Based Syst 21 (4):305–320
10. Borges A et al (2013) Ontologies supporting the distributed software development: a systematic mapping study, Proceedings of the 17th international conference on evaluation and assessment in software engineering. ACM, Porto de Galinhas, pp 153–164
11. Wongthongtham P et al (2006) Ontology-based multi-site software development methodology and tools. J Syst Archit 52(11):640–653
12. Bourque P (2003) SWEBOK guide call for reviewers. 29 May 2003. Available from: http://serl.cs.colorado.edu/~serl/seworld/database/3552.html
13. Carmel E, Agarwal R (2001) Tactical approaches for alleviating distance in global software development. IEEE Softw 18(2):22–29
14. Herbsleb JD et al (2001) An empirical study of global software development: distance and speed. In: 23rd International Conference on Software Engineering (ICSE'01). Toronto, Canada
15. Herbsleb JD, Moitra D (2001) Global software development. IEEE Softw 18:16–20
16. Boulton C (2008) IBM's social beehive and discovery search. eWeek.com
17. Giri K (2011) Role of ontology in semantic web. DESIDOC J Libr Inf Technol 31(2):116–120
18. Paydar S, Kahani M (2011) An agent-based framework for automated testing of web-based systems. J Softw Eng Appl 4:86–94
19. Portillo-Rodríguez J et al (2014) Using agents to manage socio-technical congruence in a global software engineering project. Inf Sci 264(0):230–259
20. Lee C-S, Wang M-H (2009) Ontology-based computational intelligent multi-agent and its application to CMMI assessment. Appl Intell 30(3):203–219
21. Nunes I et al (2011) On the development of multi-agent systems product lines: a domain engineering process, in agent-oriented software engineering X. Springer, Berlin/Heidelberg, pp 125–139
22. Chira C (2007) A multi-agent approach to distributed computing. Comput Intell:43–45
23. Robillard M, Walker R, Zimmermann T (2010) Recommendation systems for software engineering. Softw IEEE 27(4):80–86
24. Ponzanelli L (2014) Holistic recommender systems for software engineering, Companion Proceedings of the 36th international conference on software engineering. ACM, Hyderabad, pp 686–689
25. Xin X et al (2013) Accurate developer recommendation for bug resolution. In: Reverse Engineering (WCRE), 2013 20th Working conference on. Koblenz, Germany
26. Begel A, Yit Phang K, Zimmermann T (2010) Codebook: discovering and exploiting relationships in software repositories. In: Software Engineering, 2010 ACM/IEEE 32nd international conference on. Cape Town, South Africa
27. Moraes A et al (2010) Recommending experts using communication history, Proceedings of the 2nd international workshop on recommendation systems for software engineering. ACM, Cape Town, pp 41–45
28. Steinmacher I, Wiese IS, Gerosa MA (2012) Recommending mentors to software project newcomers. In: Recommendation Systems for Software Engineering (RSSE), 2012 third international workshop on. Zurich, Switzerland
29. Xiang PF et al (2008) Ensemble: a recommendation tool for promoting communication in software teams, Proceedings of the 2008 international workshop on recommendation systems for software engineering. ACM, Atlanta, pp 1–1
30. Sawadsky N, Murphy GC (2011) Fishtail: from task context to source code examples, Proceedings of the 1st workshop on developing tools as plug-ins. ACM, Waikiki, pp 48–51
31. Cordeiro J, Antunes B, Gomes P (2012) Context-based recommendation to support problem solving in software development. In: Recommendation Systems for Software Engineering (RSSE), 2012 third international workshop on. Zurich, Switzerland

32. Ashok B et al (2009) DebugAdvisor: a recommender system for debugging, Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering. ACM, Amsterdam, pp 373–382
33. Borges A et al (2013) Ontologies supporting the distributed software development: a systematic mapping study. In: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering. ACM, Porto de Galinhas, Brazil
34. Lim SL et al (2013) Using web 2.0 for stakeholder analysis: stakesource and its application in ten industrial projects. In: Maalej W, Thurimella AK (eds) Managing requirements knowledge. Springer, Berlin, pp 221–242
35. Lim SL, Quercia D, Finkelstein A (2010) StakeNet: using social networks to analyse the stakeholders of large-scale software projects, Proceedings of the 32nd ACM/IEEE international conference on software engineering, 1st edn. ACM, Cape Town, pp 295–304
36. Liu L. et al. (2014) An ontology-based advisement approach for SOA design patterns. In: The 8th international conference on knowledge management in organizations. Springer, Dordrecht
37. Ovaska P, Rossi M, Marttiin P (2003) Architecture as a coordination tool in multi-site software development. Softw Process: Improv Pract 8(4):233–247
38. Hass G (2003) Configuration management principles and practice. Addison-Wesley Longman Publishing Co
39. Gracia J, Mena E (2008) Web-based measure of semantic relatedness, Web Information Systems Engineering-WISE 2008. Springer, Berlin/Heidelberg, pp 136–150
40. Parsia B, Sirin E 2004 Pellet*:* an owl dl reasoner. In: Third international semantic web conference
41. Ashraf J, Hussain OK, Hussain FK (2012) A framework for measuring ontology usage on the web. Comput J 56:1083–1101
42. Hepp M (2007) Possible ontologies: how reality constrains the development of relevant ontologies. Internet Comput 11(1):90–96
43. Brickley D, Miller L (2010) FOAF vocabulary specification 0.98. In: Namespace document. RDF and Semantic Web developer community. http://www.loeyeol.com/siteagent/xmlns.com/foaf/spec/20100809.html
44. Weibel S et al (1998) Dublin core metadata for resource discovery. Internet Engineering Task Force. Network Working Group, pp 1–8
45. Miles A, Bechhofer S (2009) SKOS simple knowledge organization system reference. Technical report, W3C
46. Breslin JG et al (2005) Towards semantically-interlinked online communities, The semantic web: research and applications. Springer, Berlin/Heidelberg, pp 500–514

Chapter 14
Investigating the Scope for Agile Project Management to Be Adopted by Higher Education Institutions

Simon P Philbin

14.1 Introduction

14.1.1 Background on Agile Project Management

The practice of agile project management [1] has been proliferating over recent years [2], and this includes application to different sectors [3] thereby extending beyond the initial technology arena. Indeed agile continues to be an emerging trend in management that is making a positive impact to projects in many organisations.

A key feature of the agile management approach is the focus on meeting the underlying business needs [4] of the organisation along with the people dimension of projects being emphasised, for instance, through joint decision-making in projects as well as joint working where possible with partners, suppliers and customer representatives. In this regard, the agile approach seeks to be implicitly more inclusive when compared to more conventional forms of process-driven project management methodologies.

Originally developed in the IT (information technology) sector, agile project management was initially seen as an alternative to the so-called waterfall methodology of project management, where IT design projects are delivered via a highly ordered linear sequence that is analogous to the action of a waterfall. Conversely, agile-related methodologies, such as Scrum [5], extreme programming [6] and DSDM (dynamic systems development method) [7], are more akin to lean management and Six Sigma practices [8]. Moreover, excessive levels of project planning are avoided, and project activities are undertaken in an iterative manner. Therefore, agile management can be viewed as a flexible approach that seeks to

S.P. Philbin (✉)
Enterprise Division, Imperial College London, South Kensington, London SW7 2PG, UK
e-mail: s.philbin@imperial.ac.uk

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_14

reduce the level of early-stage planning as well as minimizing the amount of documentation that is in place for a project.

14.1.2 Agile Applications

Agile has been adopted across a number of high-tech industrial applications [9], and this extends to governmental applications [10]. However, there are different types of organisations that also undertake projects, such as charitable organisations and academic institutions. The question therefore arises: How can agile project management techniques be adopted by higher education institutions? Moreover, should academic institutions be considering new management practices and leveraging the latest and emerging management trends from industry? Indeed some initial studies have looked into the potential application of agile techniques at academic institutions [11], where a simplified agile approach was evaluated for suitability for small teams working both in academia and in industry.

Other work has looked into the use of agile approaches to support idea generation and related creative processes at universities [12]. There has also been work reported on how agile techniques can potentially support the processes and operations in research and development (R&D) laboratories [13]. Conversely, the applicability of complementary methodologies, such as BPM (business process management), has been investigated for higher education institution (HEI) applications, with such application being found to largely rest on the university having a supportive set of organisational values and strategic intent as well as the necessary leadership and people skills to support an effective introduction of BPM [14].

Consequently, these studies would appear to indicate there is an emerging interest in applying new project management methodologies in higher education institutions, organisations that can often be very traditional in their approach to managing work-based activities. Moreover, when implementing new management approaches in such organisations, it will be important to consider the organisational landscape and the underlying culture, specifically whether a culture that supports changed working practices exists.

14.1.3 Projects at Higher Education Institutions

Universities are often large and complex organisations where there can be a significant number of management challenges as well as various types of projects delivered [15]. Challenges can include the availability of funding, competition for staff and students as well as the need to deliver various types of projects. Indeed projects are a constant feature of research activities carried out by universities, and most research initiatives are delivered according to a certain project approach, i.e. to meet schedule, budget and scope or specification requirements.

Education and teaching at universities can also be viewed through a 'project lens', such as a project to develop and deliver a new Master's degree programme or implementation of new technology to improve the quality of educational delivery. In this regard, implementation of new technology to improve educational delivery has been found to be hampered in many cases with only a minority of the projects from the study resulting in improved student learning outcomes [16].

Moreover, there is a need to ensure adequate quality levels are achieved for new educational platforms and in particular online programmes if they are to be viewed as legitimate and valuable [17]. Adopting improved management techniques, such as agile management, provides scope to support such goals and contribute to quality assurance for educational programmes especially where risk can be higher through the use of new technologies.

The exchange of knowledge by universities with partner organisations, such as through technology transfer and the commercialization of IP (intellectual property) with industrial companies [18], can be viewed as project-based activities. This is because the commercialization of foreground IP that has been generated, for instance, from a scientific research study, needs to be undertaken in a timely manner so that commercial value can be secured while appropriate partners remain motivated to take the research to market. Hence, there is a need to adhere to a required schedule in addition to the financial aspects of the commercialization as well as achieving the required specification in terms of the suitability of any contractual documentation needed to underpin the transaction. Managing research projects that are funded by industry can also involve two-way knowledge exchange that is dependent on social, commercial and process-driven factors [19], where adoption of project management techniques can make a positive impact to the performance of such projects.

Therefore, higher education institutions present an emerging opportunity for agile project management techniques to be applied in order to improve the efficiency and effectiveness of operations. There is a need for projects to be delivered, and this need extends to a growing interest in the application of new techniques and methodologies that have been successfully applied in industry. Hence, the application of agile management techniques at higher education institutions is a valid line of enquiry.

Consequently, the objectives of this chapter are to provide a review of agile project management and to explore in conceptual terms how certain agile techniques can be adopted at higher education institutions. This will be undertaken via a literature review of agile project management followed by discussion of three illustrative examples of the application of agile techniques at higher education institutions.

14.1.4 Chapter Organisation

This chapter is organised as follows. After the introduction section, there is a section that provides a summary and supporting information on agile project management. This is followed by a section on identifying the scope for agile project management at higher education institutions. The next section is focused on detailing three illustrative cases that describe how agile techniques can potentially be adopted by higher education institutions. Each case study includes an initial assessment of the system requirements for implementation of the agile technique including consideration of the appropriate ICT infrastructure. This section is followed by discussion and conclusions and finally the future work.

14.2 Agile Project Management

14.2.1 Manifesto for Agile Software Development

A key development for the agile project management movement was the publication of the *Manifesto for Agile Software Development* [20]. This document includes a number of statements that essentially capture the essence of the agile approach, and while it is focused on software development, the messages can also be applied to other applications and industries. The agile manifesto [21] is as follows:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> - *Individuals and interactions over processes and tools.*
> - *Working software over comprehensive documentation.*
> - *Customer collaboration over contract negotiation.*
> - *Responding to change over following a plan.*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

The manifesto is accompanied by 12 guiding principles [21], which are provided in Table 14.1 along with an interpretation by the author of the meaning of each principle.

The manifesto and the accompanying principles provide a useful summary of the agile management approach, and it can be observed that there are a number of themes that emerge. There is a clear emphasis on the people dimension of projects, working collaboratively together as opposed to rigidly sticking to predefined rules and procedures. There is joint working that involves team members setting the direction of the project and also joint working where possible with customers or clients as well as suppliers. Agile is therefore an inclusive approach, where issues and challenges are effectively shared across the team and all team members are able to contribute to the achievement of the project's goals.

Table 14.1 Twelve agile management principles and corresponding interpretations

No.	Agile management principle	Interpretation
1	*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software*	The preeminence of customer requirements that should be delivered as early as possible with workable solutions that create value for the customer
2	*Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage*	The ability to cope with changes, issues and risks encountered throughout the project lifecycle and to accommodate such changes for the ultimate benefit of the customers
3	*Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale*	Providing early delivery of working project outputs that have partial functionality with full functionality only provided later in the project. Essentially it is better to have a working solution than no solution at all
4	*Business people and developers must work together daily throughout the project*	The need for commercial- and business-oriented people to work side by side with technical-oriented people, thereby avoiding a stovepipe mentality arising
5	*Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation*	The overall importance of the people dimension of projects and the need to support the people working on projects through providing a trusting and open environment that encourages joint working and sharing of information across the project. Also emphasizing direct face-to-face communication when compared to other forms such as electronic communication
6	*Working software is the primary measure of progress*	Further highlighting that a working solution for a project is better than no solution at all. This underscores that a partial completion of project specification can be acceptable
7	*Agile processes promote sustainable development*	The importance of projects being undertaken in a manner that will survive into the future and will not result in negative impacts on the project environment (either locally or more widely)
8	*The sponsors, developers, and users should be able to maintain a constant pace indefinitely*	Those involved in the project need to work according to a consistent approach that can be maintained and does not result in burnout of anyone involved through excessive levels of project contributions
9	*Continuous attention to technical excellence and good design enhances agility*	In order to achieve high levels of excellence and flexibility, there is always a need to maintain a focus on technical quality for any kind of project
10	*Simplicity – the art of maximizing the amount of work not done – is essential*	Projects can be completed without excessive levels of operating procedures and

(continued)

Table 14.1 (continued)

No.	Agile management principle	Interpretation
		documentation, although there should be enough to provide adequate structure and support the planning and delivery process
11	*The best architectures, requirements, and designs emerge from self-organizing teams*	Agile methods rely on teams of people jointly taking ownership of project success and not relying solely on the coordination provided by a project manager
12	*At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly*	The importance of looking back and reflecting on achievements as well as difficulties encountered to support lessons learnt and continuous improvement across the project

The ability to adapt and an inherent flexibility are also key features, changing the course of the project to adapt to emerging challenges or changes in the external environment. The ability for project workers from different disciplines and backgrounds to work together is an important feature as well as the overall benefits from adopting a trusting and open environment through sharing knowledge across projects. Clearly, these attributes would be sensible to adopt for projects delivered in all kinds of organisations.

14.2.2 Agile Methodologies

There are a number of different versions of agile management, and the Scrum methodology [22] is used widely in industry. A key feature of the Scrum approach is the articulation of project work in terms of product features and the so-called product backlog, which can be viewed as a prioritized list of all the features to be delivered by the project (see Fig. 14.1). As depicted in this diagram and through reference to the product backlog, it is possible to develop the highest-priority features for a project first thereby leaving only low-priority features to the end of the project.

Consequently, where resources or time is no longer available as the project approaches the scheduled completion date, the project work that has not been completed will by definition be lower priority when compared to the completed work and corresponding features. A further aspect of the scheme in the diagram is the work cycle, where each feature is delivered via a time-boxed iteration (say over a week or month), and each iteration for a technology-based project would involve exploration (or design) followed by development (or engineering) and then deployment (or implementation). This iterative process is repeated for each feature according to the aforementioned prioritization process.

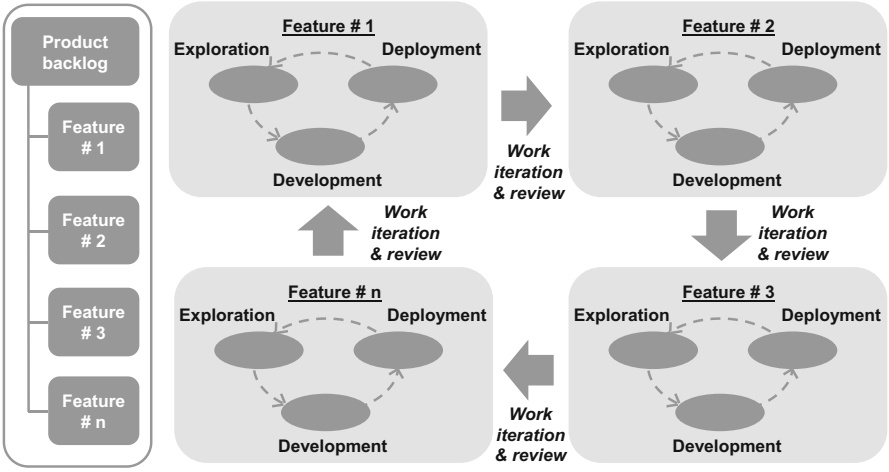


Fig. 14.1 Development cycle for agile project

Central to the agile approach is the clear focus on delivery of the project according to a fixed schedule and within the financial budget, but when needed, there can be 'flexing' or controlled modifications of the project scope. This is an implication of the sequential delivery of the higher-priority features during earlier stages in the schedule, and in cases where there is no more time remaining, only lower-priority features are not delivered. This approach does not mean quality standards are compromised, but it does mean that projects can be delivered with some working level of functionality but still within the schedule and budgetary envelopes available.

This approach fundamentally sets the agile approach apart from the traditional view of project management [23], where the so-called iron triangle of requirements have to be delivered, i.e. according to fixed schedule, financial budget and project specification. However, as many studies and reports have identified, such an approach often results in project failures and problems with delivery, for example, as reported for IT projects by the Standish Group [24]. Furthermore, construction projects continue to be delivered with time delays and cost overruns, and agile has recently been found to be a promising strategy to help cope with engineering complexity and improve the performance of such projects [25].

Agile techniques can also involve sprints over shorter timeframes, where each sprint has an interim target that must be met within a fixed 'time box' [26]. Breaking down projects into such sprints and corresponding 'time boxes' represents another approach to ensure projects are delivered within the required schedule. Although agile is an overall approach to projects, it is characterized by a wide selection of techniques that when employed collectively support the achievement of agile working according to the previously defined principles. A number of these techniques associated with agile project management will be explored further in the illustrative case studies that are discussed in this chapter.

14.3 Identifying the Scope for Agile Management at HEIs

14.3.1 Organisational Landscape for HEIs

Higher education institutions (HEIs) are complex organisations, often large in size spanning academic departments and various support services along with a wide array of stakeholders to serve. Universities need to remain relevant if they are to continue to attract high-quality students and staff. They need to respond to emerging needs and trends, such as globalization [27], improving student engagement [28] or the current strong interest in entrepreneurial activities and related educational provision [29]. Universities are also actively adopting more commercial practices, such as business planning methodologies to support the development of strategic academic programmes [30].

Moreover, universities can be powerful engines to support economic development through the knowledge that is created as well as through producing educated and trained individuals that are able to work in knowledge-based jobs in industry and in wider society. Universities are able to generate research and technology outcomes that can be adopted by industrial companies to enable improved products, services and manufacturing processes thereby helping to improve industrial competitiveness.

Universities also face a number of challenges, such as pressure on budgets, increasing levels of competition for funding and attracting the best students and staff – increasingly on an international level. Plus, there is a need to respond to the opportunities offered through pursuing collaborative partnerships with industrial organisations [31] as well as adopting ICT (information and communications technology) to improve the teaching experience. In regard to the primary organizational strategy of most universities, it can be articulated according to three core capabilities that are education, research and knowledge exchange and as depicted in Fig. 14.2.

14.3.2 Supporting the HEI Strategic Agenda

Adoption of new management systems at HEIs will therefore need to be able to make a positive impact on a university's ability to pursue a successful strategic trajectory that generates value across all three core capabilities [32]. Consequently, it is useful to explore where there is scope for agile project management to result in improvements in these three areas, and a useful framework to adopt is the so-called four E's, i.e. assessing performance in regard to improvements in efficiency, effectiveness, economy and ethical considerations [33]. Therefore, Table 14.2 provides the results of this assessment, which highlights that there is significant scope for agile practices to make a positive impact to the performance of universities across the education, research and knowledge-exchange domains.

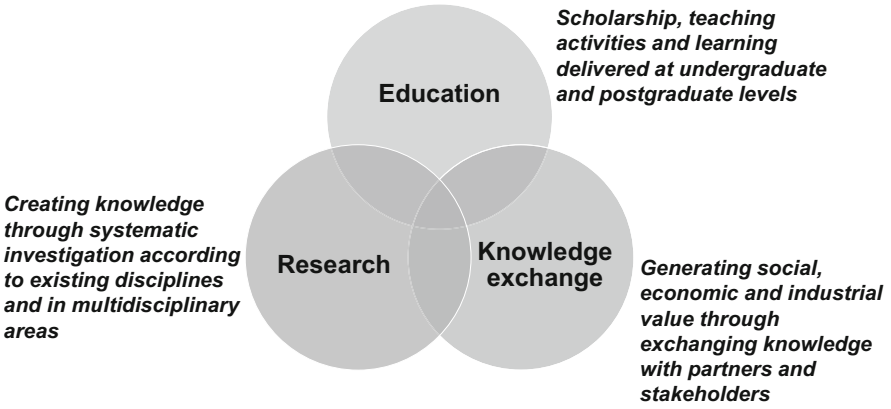


Fig. 14.2 Higher education institution core capabilities

Table 14.2 Scope for application of agile to academic institution core capabilities

Performance	Education	Research	Knowledge exchange
Efficiency	Maximising the value and quality of knowledge and skills transferred to students through optimal delivery of teaching and tuition services	According to a given level of funding, generating the optimal level of knowledge from research activities that are delivered	Translation of research outcomes to enable the optimal level of value creation in terms of industrial, societal or knowledge impact
Effectiveness	Generating improved skills and knowledge of the students through ensuring scholarship is fit for purpose and regularly updated	Achieving knowledge goals from research conducted in regard to the required deliverables, milestones and quality levels	Two-way exchange of knowledge with partners and stakeholders undertaken according to defined plans and against specified key performance indicators
Economy	Delivery of teaching and scholarly work according to a minimized cost base while still at appropriate quality levels	Research projects delivered according to required performance and quality levels, while costs are minimized as appropriate	Ensuring the transaction and follow-on costs for research translation activities with appropriate partners are minimized
Ethics	Adopting a transparent and honest approach to teaching work that is applied consistently to student cohorts along with fairness and integrity	Research conducted according to probity, integrity and ethical dimensions while avoiding any conflicts of interest that could conceivably arise	A consistent approach used to the exchange of knowledge with partners that is based on integrity, probity and diligence of application

14.4 Evaluation of Agile Techniques Through Illustrative Cases

The previous assessment exercise has highlighted the scope for agile practices to be potentially applied at universities, and it is useful to examine specific instantiations in order to provide context as well as an improved understanding of the outcomes of such an application. Therefore, three illustrative cases are provided in this chapter that highlight how agile techniques can be applied across education, research and knowledge-exchange provision at higher education institutions.

The case studies have been developed through drawing on the author's experience in research and technology projects at universities. Supporting material from the literature has been considered for each case, and through a process of reflective inquiry [34], the key findings from the cases have been synthesized. Each case also includes identification of the proposed system requirements for the corresponding agile application.

14.4.1 Case 1: Development of Online Master's Degree Programme

The development of a new online Master's degree programme can be a challenging initiative [35] for universities and the staff involved as there is a significant amount of planning required across a range of underlying areas, including consideration of blended learning options [36]. This includes making a supporting case to secure the necessary approvals within the university, ensuring the programme is intellectually demanding and designing the programme so that it provides an appropriate balance between theoretical and practical aspects where appropriate as well as making sure that the degree will be appealing to the targeted student cohorts. There is also a need to ensure the learning materials, teaching notes and scholarly content are prepared in a timely manner and for the appropriate ICT infrastructure to be available to enable delivery of the online programme.

Development of all this material in a timely fashion represents a significant challenge that would potentially benefit from the adoption of the agile Scrum approach and in particular the sprint planning technique [37]. Basically, a sprint is a fixed period of time where specified work has to be completed in preparation for review and acceptance of the project work. Each sprint period begins with a planning meeting, where the development team needs to agree on what can be realistically achieved in the given timeframe and the relevant product owner will need to make a final decision on the review criteria that need to be met in order for the work to be approved and accepted.

The sprint planning approach can be undertaken through assembling a task tracking table, and in terms of project delivery, a sprint burndown chart is used to track development of the project in real time according to achievement of the

Table 14.3 Task tracking table for online Master's degree programme development project

Task #	Task description	Task status	Task owner	Projected days	Remaining days
01	Development of programme specification	Closed	Course director (professor)	5	0
02	Programme proposal and business plan	In work	Course deputy director (lecturer)	15	5
03	External assessor's report commissioned and delivered	Open	Course director (professor)	10	10
04	Preparation of degree regulations	In work	Departmental administrator	10	5
05	Degree programme structure with details of compulsory and elective modules	In work	Course deputy director (lecturer)	25	5
06	Preparation of outline teaching materials prepared	In work	Course director (professor)	35	20
07	Information and communications technology (ICT) infrastructure development	Open	Departmental administrator	10	10
08	University approvals, i.e. at department, faculty and senate levels	Open	Course director (professor)	10	10

Note on project task status:
Open: task not started
In work: task started but not completed
Closed: task completed

required tasks [38]. Consequently, Table 14.3 provides a task tracking table, and Fig. 14.3 represents a corresponding sprint burndown chart for the illustrative development of an online Master's degree programme.

The task tracking table captures key information and can be updated throughout the term of the project, including information on the task description, status (either open, closed or in work), task owner, projected days for the task and remaining days for the task. The table is maintained throughout the project and can be used to readily track tasks and the remaining time to the completion of the project.

In regard to progress of the project, it can be seen from the table that there is one closed task, four in work tasks and three open tasks. Furthermore, the sprint burndown chart plots a linear trend line of tasks remaining (for this case, there is a total of 120 project working days over a 12-week project duration) and also the actual tasks remaining, which are recorded in real time. In the example, the sprint burndown chart is at the end of week 6, and it can be observed that the actual tasks remaining amount to 65 days' worth of work, which is a summation of the remaining days for tasks 01 through 08. In this example 55 days of work has been completed on the project.

Using this agile approach would ensure there is a clear and transparent view on the progress of the education project to develop a new online Master's degree programme, thereby making sure all the task owners (namely, the course director,

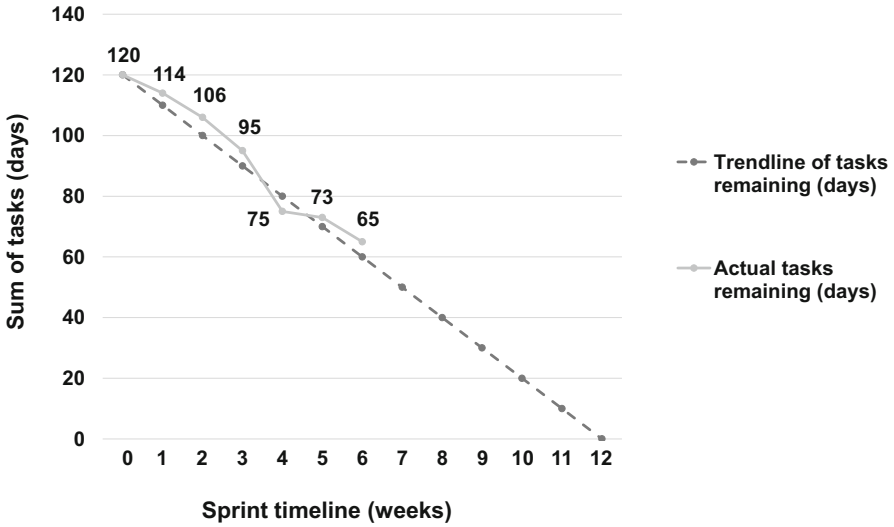


Fig. 14.3 Sprint burndown chart for online Master's degree programme development project

deputy director and departmental administrator) are updated on progress. The impact of issues and risks can be readily observed and managed through the tracking process, and use of a sprint burndown chart would allow any significant variations to the trend line to be observed so that corrective action can be undertaken rapidly to maintain the performance of the project and meet the required schedule.

In terms of ICT design and implementation, it is possible to synthesize the system requirements (SR1/1–6) [39] for this application of agile project management to the development of an online Master's programme, which are summarized as follows:

- SR1/1: The work-based information system will allow input by users of the initial task-related data (qualitative and quantitative) according to the predefined data fields in the task tracking table.
- SR1/2: The work-based information system will allow subsequent input by users of the remaining days' data field for each given task. Such input is likely to be on a periodic basis, e.g. every week.
- SR1/3: The work-based information system will automatically generate a graphical output based on a sprint burndown chart according to the sum of tasks (trend line of tasks remaining and actual tasks remaining) versus the sprint timeline.
- SR1/4: The work-based information system will allow users to assess the variation between the trend line of tasks remaining and the actual tasks remaining on a periodic basis, e.g. every week.
- SR1/5: Users will be able to engage with the work-based information system via traditional hardware interfaces (keyboard and mouse) in addition to mobile devices, thereby increasing functionality. App-based implementation is also recommended.

- SR1/6: The work-based information system to be available via cloud-based applications with scope for integration with other ICT project management applications.

This set of initial system requirements highlights the data and user aspects, interfaces as well as hardware configurations. Further refinement of these requirements is suggested through use of a more formalized requirements capture process with targeted users. Nevertheless it is useful to establish these system requirements so that practitioners contemplating adopting agile techniques can have an improved appreciation of the areas to be considered when designing and implementing the agile management systems.

14.4.2 Case 2: Multidisciplinary Medical Research Project

Medical research projects generally require contributions from different academic disciplines and are by nature multidisciplinary, for example, the development of a drug compound involving testing of an NCE (new chemical entity) [40] along with pharmacological and biochemical research that is required before the eventual testing is carried out. Open communication across multidisciplinary teams is a key factor that determines the success of collaborative research projects [41], and remote working of researchers can also be a feature.

This could, for instance, be associated with participation in international collaborative projects such as Horizon 2020 consortium projects funded by the European Commission, for example, involving health systems and policy research [42] or construction management and building materials research [43]. Indeed, the ability of the principal investigator and project researchers to work together in a collaborative fashion will likely have a significant impact on the performance of such a research project. Therefore, adopting agile processes and systems that support optimized communications as part of collaborative working will ensure data and information are shared in an open and efficient manner.

Agile project management is geared towards supporting transparent communications across the project, and a technique that can be adopted is the so-called Kanban board [44], which is a visual representation of project status that is placed on the wall so that members of the project team can become rapidly appraised of the status of the project and issues that need to be addressed.

Where team members are working remotely, this concept can be delivered through a digital information radiator, which provides real-time data and information on the status of the project, such as the product backlog, identified tasks as well as complete and incomplete assignments [45]. Consequently, Fig. 14.4 provides an illustrative representation of a digital information radiator for a multidisciplinary medical research project undertaken by a university.

The example provided includes illustrative update information that would be entered into an appropriate ICT system in real time and be accessible by project

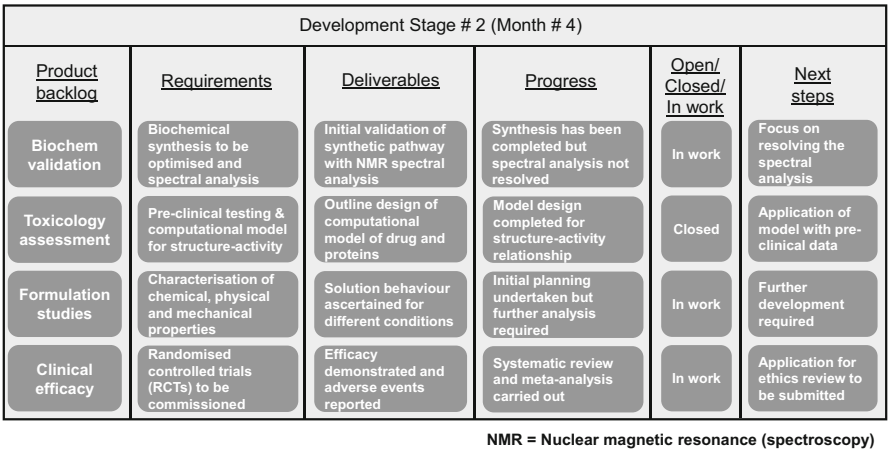


Fig. 14.4 Representation of digital information radiator for medical research project

team members wherever they are located, i.e. remotely. In the example, there are four products being worked on during development stage 2 (month 4), which are biochemical validation, toxicology testing, formulation studies and clinical efficacy. The toxicology testing product has been completed and is closed, whereas the other products remain in work with further activities to be carried out. The information radiator allows the requirements and deliverables to be easily interpreted along with the progress and next steps recorded in real time for each of the products.

In the example, it can therefore be observed that from the 'in work' products, the biochemical validation product is nearly complete, although the formulation studies and clinical efficacy products still have a significant amount of project work remaining in order to be completed. Adoption of this agile methodology would therefore enhance knowledge availability across the medical research project and for use by the participants (namely, the principal investigator, researchers and postgraduate students), thereby supporting an open and trust-based environment that is a key feature to the success of multidisciplinary research collaborations.

In terms of ICT design and implementation, it is possible to synthesize the system requirements (SR2/1–6) for this application of agile project management to the multidisciplinary medical research project, which are summarized as follows:

- SR2/1: The work-based information system will allow input by users of the product backlog-related data (qualitative and quantitative) according to the predefined data fields in the digital information radiator.
- SR2/2: The work-based information system will allow subsequent input by users according to the updated data fields for each product backlog, namely, progress, status (open/closed/in work) and next steps. Such data input is likely to be in real time or when the situation on the project has changed.

- SR2/3: The work-based information system will provide a readily accessible visual overview of the key data and information for the product backlog thereby supporting real-time project communications.
- SR2/4: The work-based information system will allow different users from across the project to input data and where appropriate support joint working with client and partner-based users. Configuration control and version control will also need to be implemented.
- SR2/5: Users will be able to engage with the work-based information system via traditional hardware interfaces (keyboard and mouse) in addition to mobile devices, thereby increasing functionality. App-based implementation is also recommended.
- SR2/6: The work-based information system will be available via cloud-based applications with scope for integration with other ICT project management applications.

As before, this set of initial system requirements highlights the data and user aspects, interfaces as well as hardware configurations. Further refinement of these requirements is also suggested.

14.4.3 Case 3: Negotiation of Industry-Funded Research Agreement

Universities are able to undertake a range of knowledge-exchange activities with industrial companies, including licensing of intellectual property (IP) [46] and working on industry-funded research projects [47]. These projects are required to generate research and technology outputs that can be transferred to industry to help improve industrial products and services or support the development of new manufacturing processes.

Both universities and companies are able to secure benefits from working together including those associated with knowledge, resources and economic aspects. However, the negotiation of industry-funded research agreements can sometimes be a particularly lengthy process due to the university and the company often having different commercial perspectives, such as those regarding the allocation of IPR (intellectual property rights). Indeed, adopting process methodologies can support the development and management of university-industry research collaborations, and it is useful to consider whether agile techniques can be applied to the negotiation of industry-funded research agreements.

Timeboxing is an agile technique [48] that involves breaking up tasks into smaller parts that can be closely managed to ensure the project schedule is met. The small parts are managed according to fixed periods of time with each period corresponding to a sprint or iteration. In a software project, a typical sprint could be 30 days in length, but this duration may be shorter for more intensive projects. Timeboxing is a technique focused on ensuring project schedules are kept under

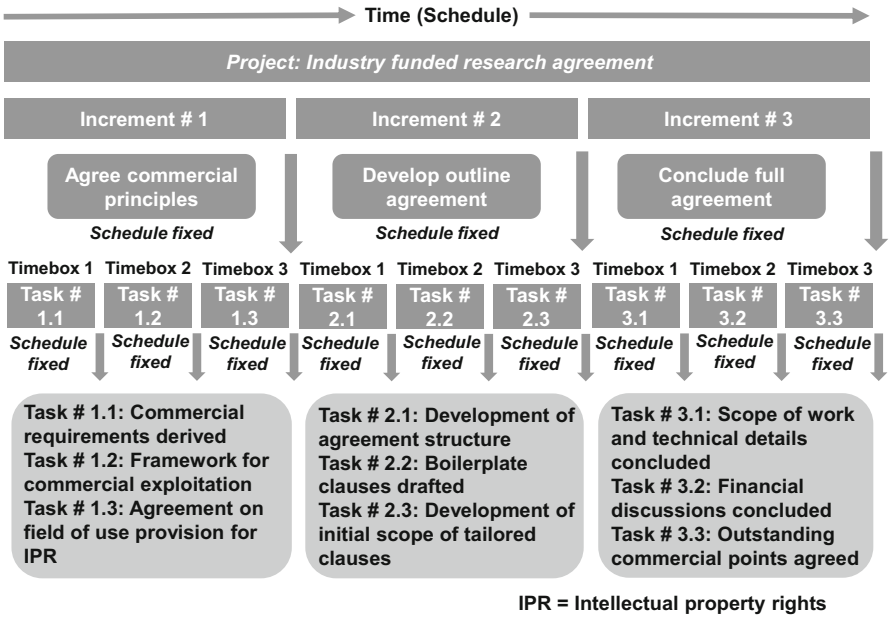


Fig. 14.5 Use of timeboxing for industry-funded research agreement project

control with the project features delivered according to the specification. Consequently, Fig. 14.5 provides an illustration of the timeboxing technique employed in support of the negotiation of an industry-funded research agreement.

In the example, negotiation of the research agreement is considered in regard to three main increments, which are as follows: agree commercial principles, develop outline agreement and conclude full agreement. Each of the increments is further subdivided into three main tasks and corresponding time boxes that are to be completed according to the fixed end dates. In the case provided, each time box could be say 4 working days long meaning that each increment would be 12 working days long, and the overall negotiation project would therefore be 36 days long.

The tasks that are allocated to each increment would be ascertained through an initial meeting of the project team involving the principal investigator (academic), university contracts manager as well as the technical and legal representatives from the company. The project team would establish the overall commercial framework from the outset and then jointly work towards achievement of the tasks within the required time. There would of course be commercial sensitivities between the two organisations associated with contractual negotiation, but nevertheless fostering an open communication channel through initial face-to-face meetings (backed up by regular video teleconferences) is more likely to result in any major contractual issues being tackled in a productive and positive manner.

Furthermore, managing the negotiation process for the research agreement would require the key tasks to be carried out according to the fixed schedule, and

as long as the 'must haves' are completed for each time box, the project should remain on schedule. 'Must haves' are the requirements that basically must be delivered for the project to be successful, as opposed to 'should haves', 'could haves' and 'won't haves', which can be developed for a project using the MoSCoW technique [49]. This is an approach to help prioritize tasks and features for a project and would be applied to help define the tasks and time boxes for the industry-funded research agreement project.

In terms of ICT design and implementation, it is possible to synthesize the system requirements (SR3/1–6) for this application of agile project management to the negotiation of an industry-funded research agreement, which are summarized as follows:

- SR3/1: The work-based information system will allow input by users of the project schedule data (qualitative and quantitative) including increments, time boxes and tasks aligned to the schedule and according to the project plan.
- SR3/2: The work-based information system will allow subsequent input by users of the status of the time boxes and tasks, including percentage completion data at given points in the schedule.
- SR3/3: The work-based information system will allow users to be able to rapidly determine via a visual interface the status of the project in terms of task completion according to the overall schedule.
- SR3/4: The work-based information system will also allow appropriate flexing or controlled modification of the tasks so as to support delivery of the project according to the fixed schedule but only where such flexing does not diminish the quality of the project outputs.
- SR3/5: Users will be able to engage with the work-based information system via traditional hardware interfaces (keyboard and mouse) in addition to mobile devices, thereby increasing functionality. App-based implementation is also recommended.
- SR3/6: The work-based information system will be available via cloud-based applications with scope for integration with other ICT project management applications.

As before, this set of initial system requirements highlights the data and user aspects, interfaces as well as hardware configurations. Further refinement of these requirements is also suggested.

14.5 Discussion and Conclusions

The implementation of agile project management is gathering pace across many organisations including both industrial companies and governmental agencies. Indeed many different types of organisations are increasingly dependent on projects to be delivered in order for operational and strategic performance to be maintained, and this includes academic institutions. Universities also face a number of

challenges and opportunities, not least through responding to the availability of new technologies as well as adapting emerging management approaches and methodologies to the not-for-profit (NFP) environment. Therefore, the question arises: Can agile project management be adopted at higher education institutions? The simple answer is yes, and this chapter has explored how this can be achieved.

Universities often rely on projects to support delivery of outputs in terms of research, education and knowledge-exchange activities. Indeed, there is much scope for agile to make a positive impact at universities in regard to performance improvements across efficiency, effectiveness, economic and ethical considerations. It is recognized that universities are places where creativity needs to be promoted and new ideas nurtured, not necessarily a place where new management practices are readily adopted. But nevertheless the opportunity to adopt agile can be viewed in the context of the general trend by universities to implement management frameworks while maintaining necessary freedoms across research and education provision [50] as well as recognising the need for greater levels of accountability [51] in the work that is delivered by universities to meet wider societal needs.

Agile management promotes a number of overriding themes, such as the importance of people working together and avoiding excessive use of standard operating procedures and unnecessarily long documentation. The ability, where and when possible, to work jointly with the customer is important as well as delivery of a working solution as opposed to holding out for the perfect solution. Originally developed in the IT sector as an alternative to the waterfall methodology, agile project management is accompanied by a raft of techniques that can be deployed to support the achievement of the agile way of working. Such techniques and related methodologies include Scrum and sprint planning, task tracking and sprint burndown charts, Kanban boards and digital information radiators, incremental management and time boxes and MoSCoW.

These techniques have been applied to three illustrative case studies through drawing on the author's experience in the higher education sector. The cases have included the development of an online Master's degree programme, a multidisciplinary medical research project and the negotiation of an industry-funded research agreement. The cases have also allowed corresponding sets of system requirements to be developed for each case instantiation. These system requirements include definition of the data entry fields (both quantitative and qualitative), baseline and update data fields, tabular and graphical interfaces, visual interfaces as well as hardware and software configurations, including remote working/app-based implementations. These requirements can be further refined through more detailed integrated system design work, and this can be supported by further requirements capture to ensure user needs are fully documented.

Although the research reported in this chapter is conceptual in nature, the illustrative cases have highlighted a number of findings and insights. Core to the agile approach is the importance of communication across projects, which can even represent a new mindset of working on projects [52]. The multidisciplinary medical research project benefits from transparent communication via the use of a digital information radiator that is accessible by collaborative team members working

remotely. The development of the online degree programme benefits from rapid communication of the task status through the sprint burndown chart and the remaining work to be completed thereby supporting development of the programme by the faculty members and administration staff. Similarly, negotiation of the industry research agreement benefits from transparent communication between the university and the company. This allows achievement of the required tasks according to the specified time boxes thereby enabling negotiation of the research agreement to be completed in the required timeframe.

Communication is a two-way process; indeed there needs to be a transmitter and a receiver, and both parties need to be responsive to open communications. Adoption of agile management practices at higher education institutions has the capacity to support those who work at universities as well as partners and funders to exercise and participate in more open forms of communication. Ultimately this enables knowledge sharing and trust-based working, which can impact enormously on the performance of higher education institutions.

It is recognized that a supporting culture will be needed to facilitate adoption of agile practices and an ability of the organization to embrace change management initiatives. The role of senior management will be important in this regard in order to drive through new working practices and ensure appropriate buy-in and adoption of agile techniques and related systems by staff members. The challenges for such implementations should not be underestimated, and any agile implementation should therefore be adapted to the local organizational context as appropriate. System implementations will need to be based on a robust understanding of the user requirements for agile working (for instance, enabling joint working by suppliers and clients as well as remote working), and where possible any agile-based systems will need to be integrated with other existing systems in use by the organization, such as an ERP (enterprise resource planning) system. Moreover, it is recognized that there are still be many instances where traditional management and close adherence to standard operating procedures (SOPs) may still be required (such as the need to accommodate government legislation on employment relations), but equally the research reported in this chapter has highlighted that there is significant scope for the adoption of agile working at higher education institutions.

14.6 Future Work

Future work is suggested to enable a more detailed examination of the implementation of agile project management at universities to build on these findings. This could involve comparative studies of agile adoption alongside other management frameworks to highlight the benefits of the more flexible and adaptable agile approach.

Other studies may involve implementation of agile management at universities followed by the use of a survey instrument to support quantitative analysis of the findings and further development of the agile research agenda. As mentioned

previously, more detailed work to capture user requirements and inform system design studies for implementation of agile management on ICT infrastructure is also recommended.

References

1. Highsmith J (2009) Agile project management: creating innovative products. Pearson Education, Upper Saddle River
2. Serrador P, Pinto JK (2015) Does agile work? – a quantitative analysis of agile project success. Int J Proj Manag 33(5):1040–1051
3. Denning S (2013) Why agile can be a game changer for managing continuous innovation in many industries. Strateg Leadersh 41(2):5–11
4. Moran A (2015) Agile project management. In: Managing agile. Strategy, Implementation, Organisation and People. Springer, Zurich, Switzerland, pp 71–101
5. Pries KH, Quigley JM (2010). Scrum project management. CRC Press, Taylor and Francis Group, Boca Raton, FL
6. Beck K (2000) Extreme programming explained: embrace change. Addison-Wesley Professional, Reading
7. Stapleton J (1997) DSDM, dynamic systems development method: the method in practice. Cambridge University Press, Cambridge
8. Salah S, Rahim A, Carretero JA (2010) The integration of Six Sigma and lean management. Int J Lean Six Sigma 1(3):249–274
9. Conforto EC, Amaral DC (2016) Agile project management and stage-gate model – a hybrid framework for technology-based companies. J Eng Technol Manag 40:1–14
10. Wernham B (2012) Agile project management for government. Maitland and Strong, London
11. Nicholls GM, Lewis NA, Eschenbach T (2015) Determining when simplified agile project management is right for small teams. Eng Manag J 27(1):3–10
12. Twidale MB, & Nichols DM (2013) Agile methods for agile Universities. In: Re-imagining the Creative University for the 21st century. Sense Publishers, Rotterdam, pp 27–48
13. Pereira IM, de Senna Carneiro TG, Pereira RR (2013) Developing innovative software in Brazilian public universities: tailoring agile processes to the reality of research and development laboratories. In: Proceedings of the 4th Annual Conference on Software Engineering and Applications (SEA 2013), Thailand, Bangkok, Global Science and Technology Forum (GSTF)
14. Mattila M, Mattila A (2014) Business process management in the context of a higher education institution. In: Proceedings in EIIC-The 3rd Electronic International Interdisciplinary conference, EDIS – Publishing Institution of the University of Zilina, vol. 3, issue 1
15. Riol H, Thuillier D (2015) Project management for academic research projects: balancing structure and flexibility. Int J Proj Organ Manag 7(3):251–269
16. Alexander S (1999) An evaluation of innovative projects involving communication and information technology in higher education. High Educ Res Dev 18(2):173–183
17. Davis JF, Sauber MH, Edwards EA (2011) Managing quality in online education: a conceptual model for program development and improvement. Int J Manag Educ 5(1):93–108
18. Powers JB (2003) Commercializing academic research: resource effects on performance of university technology transfer. J High Educ 74(1):26–50
19. Philbin S (2008) Process model for university-industry research collaboration. Eur J Innov Manag 11(4):488–521
20. Fowler M, Highsmith J (2001) The agile manifesto. Softw Dev 9(8):28–35
21. Agile Alliance (2001) Manifesto for agile software development, http://agilemanifesto.org/. Accessed 30 Nov 2016

22. Rubin KS (2012) Essential scrum: a practical guide to the most popular agile process. Addison-Wesley, Upper Saddle River
23. Highsmith J, Cockburn A (2001) Agile software development: the business of innovation. Computer 34(9):120–127
24. Standish Group (1995 and 2009) The CHAOS report. http://www.standishgroup.com/. Accessed 30 Nov 2016
25. Sohi AJ, Hertogh M, Bosch-Rekveldt M, Blom R (2016) Does lean & agile project management help coping with project complexity? Procedia-Soc Behav Sci 226:252–259
26. Han J, Ma Y (2015) Software project planning using agile. In: Progress in systems engineering. Springer International Publishing Switzerland, pp 333–338
27. Deem R (2001) Globalisation, new managerialism, academic capitalism and entrepreneurialism in Universities: is the local dimension still important? Comp Educ 37(1):7–20
28. Nelson KJ, Clarke JA, Stoodley ID, Creagh TA (2014) Establishing a framework for transforming student engagement, success and retention in higher education institutions [Final Report]. Office for Learning and Teaching, Sydney
29. Russell R, Atchison M, Brooks R (2008) Business plan competitions in tertiary institutions: encouraging entrepreneurship education. J High Educ Policy Manag 30(2):123–138
30. Philbin SP, Mallo CA (2016) Business planning methodology to support the development of strategic academic programmes. J Res Admin 47(1):23–39
31. Harman G (2005) Australian social scientists and transition to a more commercial university environment. High Educ Res Dev 24(1):79–94
32. Philbin SP (2015) Exploring the application of agile management practices to higher education institutions. In: Proceedings of the International Annual Conference of the American Society for Engineering Management (ASEM), Indianapolis, USA
33. Norman-Major K (2011) Balancing the four Es; or can we achieve equity for social equity in public administration? J Publ Aff Educ 17:233–252
34. Schön D (1983) The reflective practitioner: how professionals think in action. Basic Books Inc., New York
35. Waugh M, DeMaria M, Trovinger D (2011) Starting an online MS degree program in instructional technology: lessons learned. Q Rev Dist Educ 12(1):63
36. Singh H (2003) Building effective blended learning programs. Educ Technol-Saddle Brook Then Englewood Cliffs NJ 43(6):51–54
37. Paasivaara M, Durasiewicz S, Lassenius C (2008) Distributed agile development: using Scrum in a large project. In 2008 IEEE International Conference on Global Software Engineering Bangalore, India. IEEE, pp 87–95
38. Patanakul P, Henry J, Leach JA, Martinelli RJ, Milosevic DZ (2016) Agile project execution. In: Project management toolbox: tools and techniques for the practicing project manager, 2nd edn. Wiley, Hoboken, pp 301–322
39. Loucopoulos P, Karakostas V (1995) System requirements engineering. McGraw-Hill Inc., London
40. Shah RR (2002) Drug-induced prolongation of the QT interval: regulatory dilemmas and implications for approval and labelling of a new chemical entity. Fundam Clin Pharmacol 16 (2):147–156
41. George S, Thomas J (2015) Managing research in large collaborative teams. In: Designs, methods and practices for research of project management. Gower Publishing Limited, Surrey, pp 301–314
42. Walshe K, McKee M, McCarthy M, Groenewegen P, Hansen J, Figueras J, Ricciardi W (2013) Health systems and policy research in Europe: Horizon 2020. Lancet 382(9893):668–669
43. Pacheco-Torgal F (2014) Eco-efficient construction and building materials research under the EU Framework Programme Horizon 2020. Constr Build Mater 51:151–162
44. Corona E, Pani FE (2012) An investigation of approaches to set up a Kanban board, and of tools to manage it. In: Proceedings of the 11th International Conference on Telecommunications and

Informatics (TELEINFO'12) and the 11th International Conference on Signal Processing (SIP'12), Saint Malo, France, pp 7–9
45. Hannola L, Friman J, Niemimuukko J (2013) Application of agile methods in the innovation process. Int J Bus Innov Res 7(1):84–98
46. Thursby JG, Kemp S (2002) Growth and productive efficiency of university intellectual property licensing. Res Policy 31(1):109–124
47. Philbin SP (2012) Resource-based view of university-industry research collaboration. In 2012 Proceedings of PICMET'12: Technology Management for Emerging Technologies, Vancouver, Canada. IEEE, pp 400–411
48. Muller G (2009) System and context modeling – the role of time-boxing and multi-view iteration. Syst Res Forum 3(02):139–152. World Scientific Publishing Company
49. Waters K (2009) Prioritization using Moscow. Agile Planning, 12 January 2009
50. Nickson A (2014) A qualitative case study exploring the nature of New Managerialism in UK Higher Education and its impact on individual academics' experience of doing research. J Res Admin 45(1):47
51. Patterson G (2001) The applicability of institutional goals to the university organisation. J High Educ Policy Manag 23(2):159–169
52. Sauer C, Reich BH (2009) Rethinking IT project management: evidence of a new mindset and its implications. Int J Proj Manag 27(2):182–193

Chapter 15
Software Project Management for Combined Software and Data Engineering

Seyyed M. Shah, James Welch, Jim Davies, and Jeremy Gibbons

15.1 Introduction

Software engineering is an established discipline for the systematic creation of large and complex software systems. Relatively more recent are attempts to systematise the creation and management of large data sets, in data engineering. A key feature of these disciplines is managing engineering processes using several stages that form a development *life cycle*. This leads to a methodical process for development and separation of work into modular elements. A challenge when using this approach is integrating software engineering life cycles with the wider context of the software, for example, business processes, user requirements or indeed other codependent development life cycles. This chapter is on combining software engineering with data engineering to enable better project management, foster reuse and harness best practice from two seeming disparate domains.

The presented methodology is created from experiences on the ALIGNED project, a large, interdisciplinary research project that applies state-of-the-art software and data engineering techniques for the development of real-world systems. Several use cases have been identified that are challenging due to a lack of coherence between the software and data engineering, from collection and management of anthropological research to cataloguing and exploration of jurisprudence data and machine interpretation of large encyclopaedic data sets.

In the project, several state-of-the-art tools are under development from both domains such as Booster [1], the Model Catalogue [2], RDFUnit [3], Repair Framework and Notification (RFN), Ontology Repair and Enrichment (ORE), Dacura [4], the PoolParty Confluence/JIRA Data Extractor (CJDE) [5] and External

S.M. Shah (✉) • J. Welch • J. Davies • J. Gibbons
Software Engineering Group, Department of Computer Science,
University of Oxford, Oxford, UK
e-mail: seyyed.shah@cs.ox.ac.uk

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3_15

Link Validation (ELV) and the Unified Governance Plugins (UGP) [6]. The methodology described in this paper is used as an initial framework to support the combination, development and application of these systems, to address the project goals. These tools would not ordinarily or easily be use or developed in unison, due to the differing approaches of software and data engineering.

The importance of the presented methodology is clear: by approaching software and data engineering life cycles coherently, efficient reuse of artefacts and applying established best practice from both domains become possible. This chapter discusses the issues of integrating the two fundamentally differing approaches to building systems. The data engineering approach broadly involves building tools that act on a wide range of data, while the approach taken in software engineering is to restrict and enforce the acceptable data, making these two approaches difficult to combine in practice.

There are several further reasons that make a combined methodology impractical. Each of the life cycles can have multiple stages, which can vary across life cycles. Creating a single methodology that encompasses all of these stages is problematic because there are many possible ordering and interleaving of process steps and there are a range of possible software and data engineering life cycles that a project may use. Furthermore, software engineering life cycles tend to take a prescriptive approach to software development, whereas data engineering life cycles tend to be descriptive. The two approaches also diverge on handling unexpected data, as data engineering tools tend to filter from a large range of inputs, which leads to failure-tolerant systems, while the fail-fast or fail-safe approaches of software engineering mean systems are expected to halt on unexpected data or deal with specific classes of error. These differences mean that dependencies between the two approaches can lead to duplication of effort, complexity in the project planning and incompatible systems due to the differing development philosophies.

Instead of a single monolithic methodology, this chapter presents a combined *meta-methodology*, which consists of a lightweight, descriptive method for combining and pairing of software and data engineering life cycles. The methodology consists of a matrix of synchronisation points that are defined on a project-by-project basis and informed by the opportunities and requirements for reuse between two loosely coupled life cycles.

This chapter is structured as follows: Sect. 15.2 presents an introduction to established software engineering and data engineering project life cycles. Section 15.3 contains a discussion on the major issues with a combined methodology. In Sect. 15.4, the methodology is presented in essential form, which consists of a flexible generic mechanism to track synchronisation between separate project life cycles. The tools and use cases of the ALIGNED project are analysed post hoc using the methodology, to form the initial evaluation of the methodology in Sect. 15.5. In Sect. 15.6, a discussion of related work is presented. The chapter concludes with plans to further apply, develop and evaluate the methods described.

15.2 Software and Data Engineering Project Life Cycles

There are several software engineering methodologies that could be considered for the alignment between software and data engineering. The subset presented here is not intended to be complete; instead an overview of some prominent and notable methodologies is presented.

In the waterfall model [7], the development process is seen as a series of downwards flowing, strictly linear steps towards an end software product. The ordering and kinds of steps can vary, but typically include "analysis", "design", "implementation", "validation" and "maintenance", as depicted in Fig. 15.1. The process mimics the traditional process for large-scale physical engineering projects. This methodology was first presented as an example of how not to develop software; it is notable for observing the effects of an ordered set of distinct steps for software development. Later adaptations addressed the rigidity by allowing earlier steps to be revisited. Several derivative methods have since been developed including the incremental waterfall, evolutionary model, spiral model and structured systems analysis and design method [8].

Rapid application development (RAD) [9] and prototyping promote the role of early prototyping to inform and shape further development iterations. Rapid application development was proposed in response to the cumbersome, inflexible waterfall-like techniques, but is no longer used as a stand-alone methodology. Agile methodologies include techniques such as test-driven development, extreme programming, feature-driven development and scrum. These so-called lightweight methods focus on delivering useful software frequently, closely working with users, simplicity and adaptability to change. The principles are further set out in the Agile

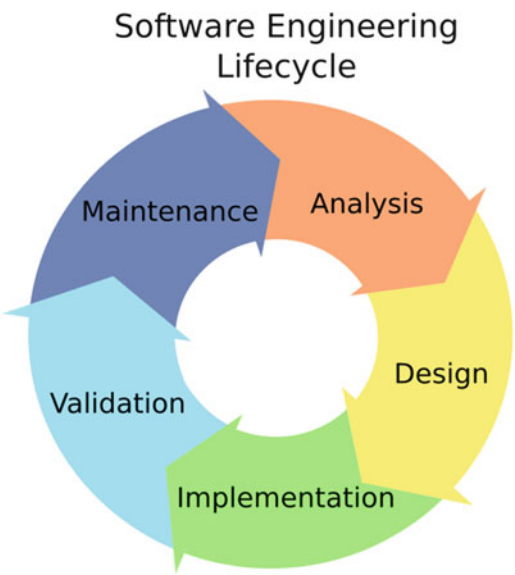


Fig. 15.1 Typical software engineering project life cycle stages

Fig. 15.2 Linked open data life cycle adapted (Adapted from Ref. [12])

Manifesto [10]. Essentially, the agile methods are designed to be reactive to the needs of the users, rather than relying on rigid one-off steps or specific languages or processes. Agile tends to afford greater trust to developers and teams to define their own processes, architectures and the selection of appropriate languages and tools.

Model-driven software development [11] revolves around the creation of high-quality, detailed models at each stage of the development process. The method relies upon the availability of sophisticated tools that use models to derive software artefacts, by a process of transformation. For example, the transformations can be used to translate designs into software implementations and models into test cases. Model-driven development aims to bring a greater level of formality to each stage of development, which can mean new tools and languages must be developed. The methodology is closely related to the techniques found in formal methods, generative programming and computer-aided software engineering. Like agile development, it does not enforce a specific set of steps, tools or languages, but instead provide a general framework within which software can be developed.

The linked open data life cycle consists of seven stages for data engineering and is described in detail by Auer et al. [12]. Essentially, the eight stages shown in Fig. 15.2 relate to the capture and management of data to ensure the data is consistent with the linked data principles. The process can begin at any stage and stages may be omitted, so the *Extract* phase can be used to take information represented in unstructured form or conforming to other structured or semi-structured formalisms and map it to the RDF data model. Once there is enough data for the intended purpose of the data, tools are required for efficient *Store and Query* of the data in triple form. The next phase is *Author* to manually create, modify and extend the structured data. As there can be many publications and

sources of information about the subjects in the store, *Link*s between data sets must be appropriately established. Since linked data relies on triples as the basic unit of data, information can be captured without the need for a high-level schema, *Enrich and Classify* is necessary to capture high-level structures, in order that data can be analysed and reusable queries can be performed. Importantly, the *Quality* of the data must be analysed and used for *Evolve and Repair*. Finally, once data has passed these phases, it should be made available for end users to *Search, Browse and Explore*, and the process can continue to refine and improve the data set.

15.3 Challenges for Projects Combining Software and Data Engineering

This chapter focuses on the alignment between software and data engineering life cycles for managing projects. This section sets out the key differences in the approaches as challenges for a combined methodology that is useful in practice.

Methodologies for software and data engineering are created for the separation of work into distinct stages during development and are sometimes referred to as life cycles, processes or methods. Each self-contained stage organises work into separate subtasks to be carried out in order to produce an end result. The stages of a methodology are usually followed linearly, and each stage results in one or more artefacts that inform subsequent stages; both are typically performed iteratively, such that the results of each iteration may be used and provide requirements for the next. Both engineering life cycles have similar goals: the management, manipulation and interpretation of data systems for eventual consumption by the end user; however, this task is approached from two divergent doctrines. Unsurprisingly, data engineering is data oriented—the data is the principal concern; in software engineering, it is the software programs that manipulate data that are the fundamental product. The software engineering life cycle results in a multi-author software artefact; in data engineering the result is a multi-origin data artefact.

While the software and data engineering life cycles produce different kinds of artefacts, there are several potential advantages of having an integrated methodology. Firstly, there is a scope for overlap in processes in each methodology. Software engineering techniques are employed to create software for data engineering (or applications that consume or manipulate the data artefacts), and data engineering processes often affect the software engineering life cycle. However, the two processes often run separately and in parallel to each other with few synchronisation points. Furthermore, artefacts produced in data engineering processes should ideally be reused in software engineering and vice versa, with reuse being a fundamental principle in both software and data engineering. This can in turn lead to software that more closely reflects the domain, and software more tailored to the data, at lower effort.

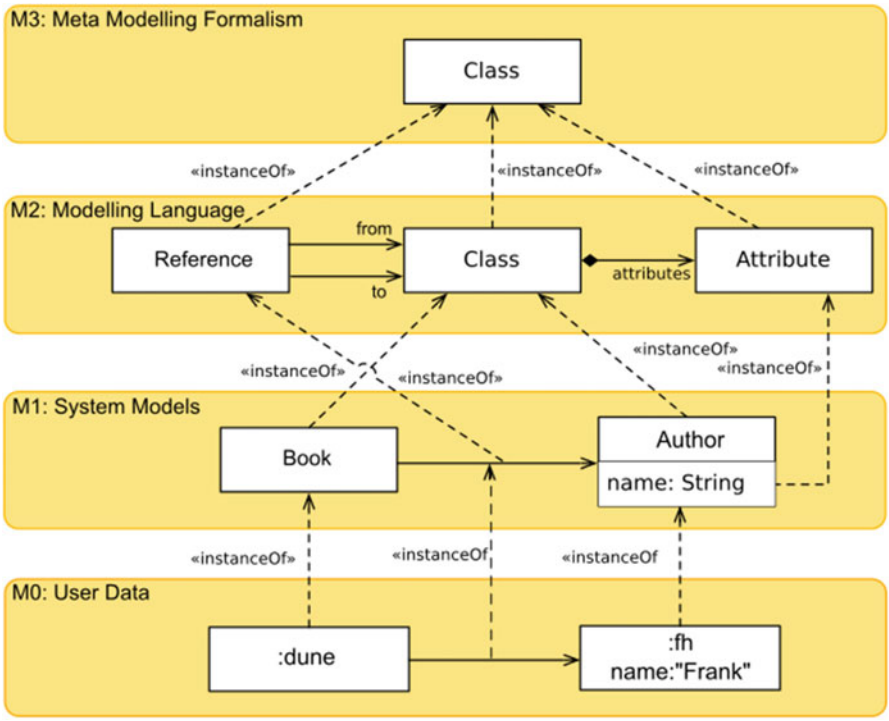


Fig. 15.3 Metalevel hierarchy in model-driven engineering

In metamodelling, data is stratified across so-called metalevels with each level used to define the primitives of the lower levels. So, instance or user data exists at level M0, which is defined in terms of the system models at level M1, which records the software and schemas to which instance data must conform. There are further metalevels, notably M2 for defining metamodels, where tools for software engineering processes are defined. Considering the metalevels, software and data engineering processes tend to be concerned with levels M1 and M0, but with differing approaches. Metalevels are a conceptual construct that simplify the definition and comprehension of complex software, by allowing separation of concerns. Existing software and data engineering data, programs and development tools can be expressed in terms of these metalevels, as shown in Fig. 15.3. Tools developed at the M2 level can be used and reused by developers at the M1 level where each new problem has common aspects that are dealt with elegantly by the M2 tools.

The approach of software engineering is to create software at the M1 level based on models that are fixed at development time and rarely change. The data that is accepted by the software is limited, and the software usually has well-defined functionality, which can involve quite complex data-dependent manipulation. This amounts to what is known as the “closed world” assumption [13]; although

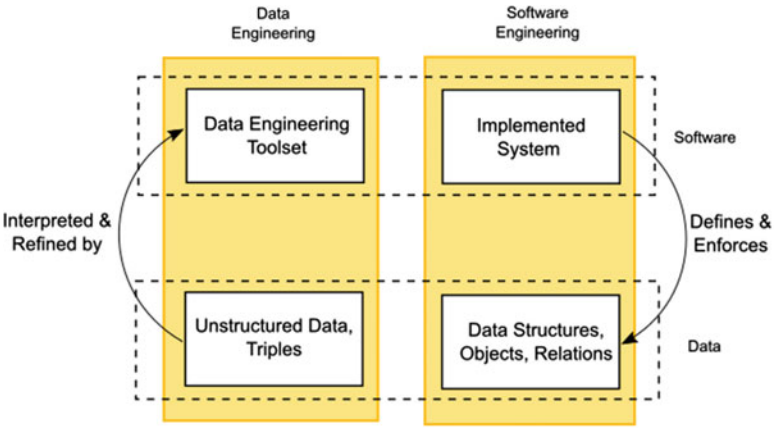


Fig. 15.4 Software and data engineering: approaches to software and data set development

it is possible to develop software without this constraint, this assumption is typical for commercial software. Tools engineered this way rely on high-quality data, and the premise is that the system should only act on fixed models and metamodels. This means that software will fail once data is encountered that does not conform to the expected model. Software engineering life cycles tend to be prescriptive with a set of procedures that must be followed in order, so a system developed using a methodology cannot be implemented until a design is in place.

Data engineering processes tend to use data-independent software tools that consume M1 level schemas, ontologies and vocabularies, to manipulate instance data in a generic way, at the M0 level. Tools to define and manipulate the schemas, ontologies and vocabularies are essential to data engineering. These tools tend to be less restrictive on the data accepted as input. Higher complexity in data processing and the ability to handle unexpected or low-quality data may require higher computational and engineering overhead; however, this usually results in tools that can be reused for unrelated data sets. The tools tend to be tolerant of low-quality data, in the sense that any unexpected data is ignored without interrupting processing. The methodology for data engineering tends to be descriptive—used more as a set of guidelines than a rigorous process, so, for example, classification and enrichment can be carried out without interlinking and fusing. The contrasting approaches of software and data engineering are shown in Fig. 15.4.

This dissimilarity between the two processes means that combining the two into a single monolithic methodology is problematic. A choice would need to be made, a priori, about prescriptive versus descriptive, fail-fast versus failure tolerant, the selection of a specific data engineering and software engineering "flavour", the ordering and interleaving of process steps and so on. This would ultimately result in a methodology that favours the perspective of either software or data engineering and may even result in a worst-of-both-worlds scenario. The methodology

presented here will not set out universal commandments, with multiple combined points of failure or a prescribed, redefined set of steps derived from either method. Instead, the aim is to try to integrate the best of both.

The methodology is based on the processes used in the ALIGNED project, where data engineering and software engineering are combined in practice. This means the methodology reflects on the observed practices and experiences of the ALIGNED consortium. This also means that the methodology may, initially at least, be specific to the tools and use cases of the ALIGNED project. Many techniques are used for data engineering and software engineering; rather than selecting and combining a single one of each, a more generic approach is suggested.

We propose the concept of an agile *meta-methodology*. This gives the software and data engineering practitioners a lightweight, basic method for combining their own methods and processes. This should, by design, be adaptable and reusable depending on the software engineering and data engineering context. The methodology provides practitioners with a system that is largely controlled by the methodology users and works practically on a project-by-project basis.

15.4 Methodology for Combined Software and Data Engineering

This section outlines the proposed *meta-methodology* for combined software and data engineering in ALIGNED. It is important to note that the present methodology is in under development and is expected to evolve and develop as the project goes on, based on the needs of the consortium and the project findings. Several areas of the method may be extended, based on observations of practice.

As a lightweight methodology, the technique requires some initial setup and maintenance by the software and data engineering processes. The main setup task for combined software and data engineering is to determine the synchronisation points between the software engineering and data engineering. A synchronisation point is at any pair of points in the two life cycles where specific artefacts and processes should be shared between software and data engineering processes. Once the synchronisation points are determined, they can be used during the project to resynchronise as development to the artefacts occurs.

In order to visualise and understand the synchronisation points between data engineering and software engineering, a table can be created that combines the two life cycles. The table allows the developers in both life cycles to determine where in each of the two life cycles synchronisation should happen. As shown in Table 15.2, the table columns are the software engineering life cycle stages, and the rows are the data engineering life cycle stages. The software engineering or data engineering life cycle stages can be used for rows or columns, as desired. Importantly, the exact stages of the software engineering or data engineering used may change depending on the data engineering or software engineering methodology used in the particular

project. The table cells represent intersections between the software and data engineering life cycles, where artefacts can be reused between life cycles, and are represented in Table 15.2 as tools that facilitate this reuse.

Before the software engineering and data engineering methodologies can be combined, terminology and equivalence between the terms must be agreed upon within a project. A standard equivalence between the generic concepts in data, model-driven software and program language engineering is shown in Table 15.1. Depending upon the scope of the project, the equivalence may not be as direct as those shown. So, for example, a particular upper ontology may be used as a model in software engineering, which may be represented at program runtime in practice, for a particular project. The abstraction level at which each artefact is expected to be used when shared between data engineering and software engineering processes should be documented as part of the process.

Table 15.2 shows an example of the incomplete synchronisation table for the ALIGNED project. The acronyms used here are expanded in more detail in Table 15.3. It uses the data engineering life cycle stages defined in the LOD2 project [12] and the software engineering life cycle stages used in the ALIGNED project. In this table, the synchronisation points indicate where data engineering tools are expected to interoperate with the software engineering processes and vice versa. Each entry in the table indicates a point that requires collaborative effort between software and data engineering processes.

Table 15.1 Comparison of terminologies in software and data engineering

Data engineering	Software engineering	Programming	Metalevel
Schema, ontology language	Meta metamodel	Grammar notation	M3
Upper ontology	Metamodel	Language grammar	M2
Domain ontology, schema	Model	Program definition	M1
Triple, data set	Instance, object	Program runtime	M0

Table 15.2 An in-progress table of the synchronisation points on the ALIGNED project

	Software engineering				
Data engineering	Analysis	Design	Implementation	Validation	Maintenance
Manual revision/ author	UGP	UGP			Dacura
Interlink/fuse					
Classify/enrich	ORE	Model Cat.			
Quality analysis	Dacura	ORE			Dacura
Evolve/repair	ORE	Dacura		RFN	
Search/browse/ explore	Model cat.				Booster
Extract	Dacura	CJDE			
Store/query			Booster		

Table 15.3 A tool-oriented synchronisation table for the ALIGNED project

	Software engineering				
Data engineering	(1) Analysis	(2) Design	(3) Implementation	(4) Validation	(5) Maintenance
(A) Manual revision/author	UGP	UGP			Dacura
(B) Interlink/fuse					
(C) Classify/enrich	ORE (enrich)	Model catalogue			
		ORE (enrich)			
(D) Quality analysis	Dacura	Dacura			
	RDFUnit	ORE			Dacura
	ORE (validation)				RDFUnit
					RFN
					ELV
(E) Evolve/repair	ORE (repair)	Dacura	RFN		RFN
		ORE			ELV
(F) Search/browse/explore	Model catalogue				Booster
(G) Extract	Dacura				
	CJDE				
(H) Store/query			Booster		

For example, the Model Catalogue tool as presented in [1] appears as a synchronisation between the design phase of the software engineering life cycle and the classify/enrich phase of the data engineering life cycle. In the context of the ALIGNED project, this means the Model Catalogue can produce an artefact (a model) that can be consumed by the data engineering "classify" phase, as a schema. In this case, the Model Catalogue can also consume schemas from the data engineering life cycle and use them as models in the software engineering life cycle. The identification of synchronisation points is very much dependent on the project and tools involved and must be done with agreement between software and data engineering processes.

The filled-in table can then be used during the project to trigger discussion and collaboration between software and data engineers. In the Model Catalogue example, this means if a new model has been created as a result using of the Model Catalogue, the model may need to be passed to the data engineering process. The exact formats and mechanism of the interchange must be decided on a case-by-case basis and requires close collaboration between the Model Catalogue developers and the schema producers to agree the format and processes for the exchange. The exchanged artefacts can include specific items such as schemas or less formal documentary artefacts such as a list of requirements.

The initial creation of the table involves at least the software and data engineering practitioners, to decide where the synchronisation points exist. There may be multiple tools at each synchronisation point, and each may have more than one mechanism to align the software and data engineering processes. However, the practitioners must at least decide the metalevel of the exchange and the model(s) at the $(m + 1)$ metalevel (from Fig. 15.3) that will be used to parse and manipulate the data. The frequency of synchronisation must also be determined; for example, a fixed schema may be exchanged once during development, so that the conforming data can be passed between running tools automatically and continuously via APIs at runtime. The data and systems' users can also be involved in the creation of the table, to identify where the requirements for the project will be met. Also data managers, curators and software users may be able to identify from the table where software and data engineering tools interact. An overview of the process is shown in Fig. 15.5.

Both life cycles are expected to run in parallel, and the aim of a combined methodology is twofold: to reduce effort and to produce artefacts that reflect best practice in both software and data engineering. However, the artefacts exchanged at any given synchronisation point may be critical to further development, which may lead to overlap in work done, bottlenecks and problematic dependencies. As engineering life cycle stages produce artefacts, there is potential for waste of effort, as artefacts may be replaced with input from an external life cycle. In part, this is mitigated by knowing which artefacts are expected and when in the life cycle, as per the table. Iteration with a high frequency can also help: synchronisation need not be final. For example, the schema need not be finalised in the data engineering process before matching software is developed—an intermediate version can be produced

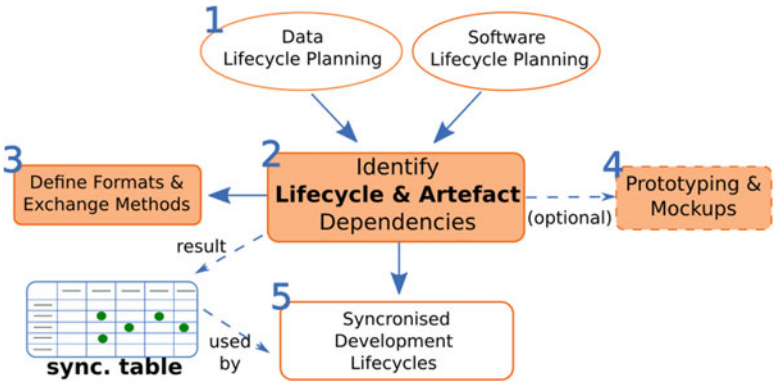


Fig. 15.5 Life cycle stages for integrated software and data engineering

so that both sides can continue working and an updated schema can be produced during the next iteration of the project life cycle.

15.5 ALIGNED Project: A Case Study for Combined Software and Data Engineering

In this section, we demonstrate how the ALIGNED project has been managed using the proposed methodology. The ALIGNED project can be seen as a concrete instantiation of the proposed *meta-methodology*. In this project, there are 28 tools with synchronisation points identified. Without the table, these points may have been overlooked or identified late. Each one of the points now has a strategy for carrying out the synchronisation. The tools and software components in the development of ALIGNED are described in the synchronisation table, below.

Table 15.3 outlines the tool-oriented view of the synchronisation between software and data engineering life cycles for ALIGNED. Each entry of the table represents a synchronisation point within the project. Note that there may be several synchronisation points per table, where multiple tools exploit reuse and best practice between domains. Also not all cells contain any synchronisation points, which reflect the scope and focus of the project. The following summary describes the high-level features of the synchronisation taking place between life cycles at each point and the development bridges between software and data engineering:

15.5.1 Manual Revision/Author (A)

- Unified Governance Plugins (UGP) (A1, A2): This tool makes use of the RDF data extracted from a software engineering support tool for bug tracking. It is

used to assist requirements gathering in the software engineering life cycle by managing documents and issues (e.g. duplicate detection or suggestions) and affects the design decisions of the system.
- Dacura (A5): In the Semantic Web maintenance phase, tools can help to identify software bugs that are caused or triggered by data-quality issues and produce notifications for authors to correct these data errors.

15.5.2 Classify/Enrich (C)

- ORE (C1): Tools for ontology validation, enrichment and repair that can provide ontology violations, suggest new ontology axioms or suggest semi-automatic fixes for the benefit of software engineers.
- Model Catalogue (C2): In the design phase of a software engineering life cycle, tools are used to create models, by applying domain expertise and reusing existing models. Models may be used as the basis for software. These schemas can be transferred from the software engineering domain to the data engineering domain as ontologies. The ontologies from the data engineering life cycle can also be transferred to the software engineering life cycle for use as the basis for creating software.
- ORE (C3): Based on the ontology analysis by ORE, changes are made to the ontology that software engineering tools can use to improve the underlying model.

15.5.3 Quality Analysis (D)

- Dacura (D1): In data engineering this tool can be used to produce data-quality tolerance requirements to constrain the data that is harvested, and these can be used as the basis for requirement in software engineering.
- RDFUnit (D1): The data engineering tools can be used to identify and analyse quality issues in the data. The results can give software engineers an insight of the underlying data set and provide feedback to tackle any quality issues.
- ORE (D1): The tool can provide ontology violations, suggest new ontology axioms (enrich) or suggest semi-automatic fixes for resolving violations. This can inform the requirement phase of software engineering, with information about data-quality issues.
- ORE (D2): Based on the ontology analysis carried out by the tool, design of the ontology can be improved.
- Dacura (D2): This defines statistical data-quality measures which must be met by the data. The tool supports the software engineering design phase by suggesting User Interface refinements to eliminate errors.

- Dacura (D5): This can help to identify software bugs that have been caused by data-quality errors and produce notifications for authors to correct these data errors and make software engineering systems more robust.
- RDFUnit (D5): In the maintenance phase, this tool can be used to identify constraint violations. Based on the analysis and design implementation, the violation results can be fed to a quality repair tool or human editors to fix the violations.
- Repair Framework and Notification (RFN) (E5): This tool checks data sets against specific data constraints (e.g. expressed as SHACL or ShEx documents) and provides semi-automatic repair strategies if violations are encountered. The tool is used in both implementation and maintenance phase because the defined data constraints influence the implementation of algorithms in the software engineering phase. Also as taxonomies are changed (additions, removals, merging with other sources), the data need to be checked for consistency with the new constraints.
- External Link Validation (ELV) (D5): This tool dereferences links from PoolParty taxonomies to "external" resources on the Web and provides statistics on invalid (broken) links. The synchronisation point is used on a regular basis for maintaining taxonomies created with PoolParty and informs the software engineering phase.

15.5.4 Evolve/Repair (E)

- ORE (E1): The tool can provide ontology violations, suggest new ontology axioms (enrich) or suggest semi-automatic fixes (for resolving violations). The analyses of the ORE findings drive the design and improvement of the underlying model in the software engineering life cycle.
- Dacura (E2): This tool can define statistical data-quality measures which must be met to support software engineering and suggest UI refinements to eliminate errors.
- ORE (E2): Based on the ontology analysis, this tool drives the design and improvement of the ontology, and this knowledge is transferred to model modifications in the software engineering phase.
- Repair Framework and Notification (RFN) (E4): The synchronisation point is used in both implementation and maintenance phase because the defined data constraints influence the implementation of algorithms, and, as taxonomies are changed (additions, removals, merging with other sources), the constraints need to be satisfied.
- Repair Framework and Notification (RFN) (E5): The tool is used in both implementation and maintenance phase because the defined data constraints influence the implementation of algorithms, and, as taxonomies are changed (additions, removals, merging with other sources), the constraints need to be satisfied. External Link Validation (ELV): This dereferences links from

PoolParty taxonomies to "external" resources on the Web and provides statistics on invalid (broken) links. The tool is used on a regular basis to suggest maintenance tasks in PoolParty.

15.5.5 Search/Browse/Explore (F)

- The Model Catalogue (F1): In the analysis phase of a model-driven software engineering, this tool is used to explore and gather metadata related to the system under construction. In the search browse and phase of the data engineering life cycle, this translates to searching and browsing the ontologies of data set.
- Booster (F5): In the maintenance phase of the software engineering life cycle, Booster can be used to create tools to extract data from a data store. The data may be used directly by users or by other tools via an API. In the data engineering context, Booster-generated tools may be used to provide a well-defined API and search data and gather data into the data store.

15.5.6 Extract (G)

- Dacura (G1): This can be used to produce data-quality tolerance requirements to constrain the data that is harvested and informs the software engineering analysis phase by defining what data is to be harvested.
- Confluence/JIRA Data Extractor (CJDE) (G1): This tool connects to software engineering management tool (Confluence/JIRA) installations (using the REST APIs) and extracts relevant requirement information and tickets (e.g. bugs, improvements) and creates RDF data. The RDF data is used to inform the data engineering life cycle.
- Confluence/JIRA Data Extractor (CJDE) (G3): This tool extracts relevant requirement information and tickets and creates RDF. This synchronisation is used on a regular basis in requirement design phase to understand the changes needed to the PoolParty tool.

15.5.7 Store/Query (H)

- Booster (H3): In the implementation phase of the software engineering life cycle, Booster-generated systems provide, create, read, update and delete functionality for data in a data store, as well as implement any user-specified actions, which can be accessed as triples via an API.

15.6 Related Work

Data-intensive systems require careful alignment between data engineering and software engineering life cycles to ensure the quality and integrity of the data. Data stored in such systems typically persists longer than, and may be more valuable than, the software itself, and so it is key that software development is sympathetic to the aims of "big data": scalability to large volumes of data; distributed, large-scale research across multiple disciplines; and complex algorithms and analysis. These are normally described in the literature as the four V's of big data: velocity, variety, volume and veracity [14].

In existing development methodologies, software and data engineering are considered as separate concerns [15]. Integrating these introduces a number of new challenges: software engineering aims of software quality, agility and development productivity may conflict with data engineering aims of data quality, data usability, and researcher productivity. Further challenges include federation of separate data sources, dynamic and automated schema evolution, multisource data harvesting, continuous data curation and revision, data reuse and the move towards unstructured/loosely structured data.

Auer et al. [12] identify challenges within the domain of life cycles for linked data projects. These include extraction, authoring, natural-language queries, automatic management of resources for linking and linked data visualisation. Typically seen as concerns for data life cycles, they all have a major impact upon software development: the authors mention component integration, the management of provenance information, abstraction to hide complexity and artefact generation from vocabularies or semantic representations.

Mattmann et al. [16] use their experience of data-intensive software systems across a range of scientific disciplines to identify seven key challenges which may be summarised as:

- *Data volume*: Scalability issues that apply not just to the hardware of the system, but may affect the tractability and usability of the data.
- *Data dissemination*: Distributed systems bring challenges of interoperability and can lead to complex system architectures.
- *Data curation*: Supporting workflows and tools for improving the quality of data, in a way that allows subsequent inspection or analysis.
- *Use of open source*: Complex technologies will depend upon reliable, reusable components supporting generic functionality.
- *Search*: Making the data collected available in a usable fashion to users, including access to related metadata.
- *Data processing and analysis*: Boiling down to workflows, tasks, workflow management systems and resource management components.
- *Information modelling*: The authors state that "the metadata should be considered as significant as the data".

The authors split these challenges into further subcategories and point out the many interdependencies between these problems. Zaveri et al. [17] take a broader view, highlighting inadequate tool support for linked data-quality engineering processes. Where tool support does exist, these tools are aimed at knowledge engineers rather than domain experts or software engineers.

Anderson [18] agrees with this issue, describing a more wide-ranging lack of support for developers of data-intensive systems. He also identifies "the necessity of a multidisciplinary team that provides expertise on a diverse set of skills and topics" as a nontechnical issue that can be addressed by projects dealing with large, distributed data sets. A technical equivalent to this issue is to understand notions of iteration with respect to the data modelling—he argues that domain knowledge is required in order to understand data collection and curation. Subsequently, he also argues for technical knowledge in order to match frameworks with requirements, emphasising the need for a multidisciplinary team.

Some solutions to these challenges have been identified—most notably in the area of model-driven software engineering, domain-specific languages and generative programming. These approaches, in combination with linked data languages and schemas, enable self-describing data structures with rich semantics included within the data itself. Aspects of program logic previously encapsulated in software are now embedded in data models, meaning that the alignment between data and software engineering becomes even more important. But these approaches can lead to further problems. Qiu et al. [19] identify two issues: firstly, there is an interaction between domain experts and application developers, and, secondly, that change to schema code may not always impact application code in a straightforward manner.

15.7 Conclusion

The proposed methodology has been put into practice and observed for utility: however, it can be seen as a hypothesis that will be refined in future iterations. One extension will record and generalise the tasks required at each synchronisation point. These observations may be condensed into software and data engineering experience "pearls" [20], for later reuse. As the project progresses, new and previously unidentified synchronisation points may be added to the table. Similarly, overlapping synchronisation points may be merged, where there is duplication of effort. Synchronisation points may also be moved or removed, depending on changes or errors discovered in the requirements of either software or data engineering project management.

Meanwhile, the methodology is to be refined by monitoring how the methodology is used within the consortium, by way of surveys and workshops. Some analysis is needed on the correspondence between the use cases and the tools developed on the project. Further analysis may be necessary for user-oriented perspectives of the methodology on the effect of the synchronisation points on the development efforts of consortium partners.

This chapter has presented a methodology for combining software and data engineering project life cycles. The hurdles to a combined methodology have been discussed and include the differences in prescriptive and descriptive philosophies and the changing needs from project to project. An agile *meta-methodology* for combined project management has been proposed as a solution to the identified issues. The benefits of the methodology are clear: reuse of artefacts between domains and application of best practice from both domains transparent project management when multiple life cycles are involved. The application of the methodology in the interdisciplinary ALIGNED project has been presented as an initial case study.

References

1. Davies J, Gibbons J, Welch J, Crichton E (2014) Model-driven engineering of information systems: 10 years and 1000 versions. Sci Comput Program 2014:88–104
2. Davies J, Gibbons J, Milward A, Milward D, Shah S, Solanki M, Welch J (2015) Domain specific modelling for clinical research. In: Proceedings of the workshop on domain-specific modeling. ACM, New York
3. Dimou A, Kontokostas D, Freudenberg M, Verborgh R, Lehmann J, Mannens E, Hellmann S, de Walle RV (2015) Assessing and refining mappings to rdf to improve dataset quality. In: Proceedings of the 14th international semantic web conference
4. Feeney KC, O'Sullivan D, Tai W, Brennan R (2014) Improving curated web-data quality with structured harvesting and assessment. Int J Semant Web Inf Syst 2014:35–62
5. Schandl T, Blumauer, A (2010) PoolParty: SKOS thesaurus management utilizing linked data. In: The semantic web: research and applications: 7th extended semantic web conference, ESWC 2010, Heraklion, May 30–June 3, 2010, Proceedings, Part II, Springer, Berlin
6. Kontokostas D, Mader C, Dirschl C, Eck K, Leuthold M, Lehmann J, Hellmann S (2016) Semantically enhanced quality assurance in the JURION business use case. In: The semantic web. Latest advances and new domains: 13th international conference, ESWC 2016, Heraklion, Proceedings, Springer, May 29–June 2 2016
7. Royce WW (1970) Managing the development of large software systems. In: Proceedings of IEEE WESCON. Los Angeles
8. Larman C, Basili VR (2003) Iterative and incremental development: a brief history. Computer 2003:47–56
9. Martin J (1991) Rapid application development. Macmillan Publishing Co., Indianapolis
10. Fowler M, Highsmith J (2001) The agile manifesto. Software Dev 9(8):28–35
11. Selic B (2003) The pragmatics of model-driven development. In: IEEE Softw. Sept 2003, pp 19–25
12. Auer S, Bühmann L, Dirschl C, Erling O, Hausenblas M, Isele R, Lehmann J, Martin M, Mendes PN, van Nuffelen B, Stadler C, Tramp S, Williams H (2012) Managing the life-cycle of linked data with the LOD2 stack. In: The semantic web – ISWC 2012: 11th International Semantic Web Conference, Boston, 11–15 Nov2012, Proceedings, Part II, Springer, Berlin/Heidelberg
13. Reiter R (1978) On closed world data bases. Springer US, Boston
14. Hitzler P, Janowicz K (2013) Linked data, big data, and the 4th paradigm. Semantic Web 2013:233–235
15. Cleve A, Mens T, Hainaut J-L (2010) Data- intensive system evolution. Computer 2010:110–112

16. Mattmann CA, Crichton DJ, Hart AF, Goodale C, Hughes JS, Kelly S, Cinquini L, Painter TH, Lazio J, Waliser D et al (2011) Architecting data-intensive software systems. Springer, New York
17. Zaveri A, Rula A, Maurino A, Pietrobon R, Lehmann J, Auer S (2015) Quality assessment for linked data: a survey. Semant Web 2015:63–93
18. Anderson, KM 2015 Embrace the challenges: software engineering in a big data world. In: Proceedings of the First International Workshop on BIG Data Software Engineering, IEEE Press, Piscataway
19. Qiu D, Li B, Su Z (2013) An empirical analysis of the co-evolution of schema and code in database applications. In: Proceedings of the 2013 9th joint meeting on foundations of software engineering. ACM, New York
20. Bentley J (1986) Programming pearls. ACM, New York

Index

Z. Mahmood (ed.), *Software Project Management for Distributed Computing*,
Computer Communications and Networks, DOI 10.1007/978-3-319-54325-3