JYVÄSKYLÄ STUDIES IN COMPUTING
269

Mervi Koivulahti-Ojala

On UML Modeling Tool Evaluation, Use and Training

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 269

Mervi Koivulahti-Ojala

On UML Modeling Tool Evaluation, Use and Training

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Delta-salissa
joulukuun 8. päivänä 2017 kello 16.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, Delta hall, on December 8, 2017 at 16 o'clock.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2017

On UML Modeling Tool Evaluation, Use and Training

JYVÄSKYLÄ STUDIES IN COMPUTING 269

Mervi Koivulahti-Ojala

On UML Modeling Tool Evaluation, Use and Training

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2017

Editors
Marja-Leena Rantalainen
Faculty of Information Technology, University of Jyväskylä
Pekka Olsbo, Ville Korkiakangas
Publishing Unit, University Library of Jyväskylä

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-7273-8

URN:ISBN:978-951-39-7273-8
ISBN 978-951-39-7273-8 (PDF)

ISBN 978-951-39-7272-1 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2017

ABSTRACT

Koivulahti-Ojala, Mervi
On UML modeling tool evaluation, use and training
Jyväskylä: University of Jyväskylä, 2011, 86 p. (+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 269)
ISBN 978-951-39-7272-1 (nid.)
ISBN 978-951-39-7273-8 (PDF)

Unified Modeling Language™ (UML) is an international standard for systems modeling. UML is used for modeling requirements, architecture, detailed design, and software code generation. UML modeling tools offer graphical editors for UML model development, generating software from UML models, creating UML models from the software, and supporting collaborative model development. This thesis offers new knowledge about UML modeling tool use, evaluation, and training. The main research question is: How can a globally distributed product company where UML modeling activities are scattered across different locations and countries implement a UML modeling tool? Five studies comprise the research process. The first study provided new information concerning how UML and UML modeling tools can be used in the context of product requirements and release management process. In the second study, version management capabilities of the UML modeling tool were evaluated. The main contribution of this study was the creation and evaluation a set of evaluation criteria. A virtual meeting tool (VMT)-based training method for teaching UML and the features of a UML modeling tool was designed and evaluated in the third study. According to the study, the VMT-based training positively impacted learners' skills, knowledge, and motivation and they were satisfied with the training. The training cost decreased in the case company by 88% compared to traditional classroom training. In the fourth study, a new instrument was developed for measuring users' satisfaction with the UML modeling tool and service. A longitudinal case study was conducted to evaluate several classes of e-teaching tools supporting the teaching of UML and the UML modeling tool during the fifth study. E-teaching tools facilitate learning both asynchronously (e.g., Wikis) and synchronously (e.g., video-conferencing). According to this study intranet and virtual meeting tool (VMT) were used to support UML and UML modeling tool training in terms of application knowledge covering commands and tools embedded in the information system; business context knowledge covering the use of information systems to effectively perform business tasks; and collaborative task knowledge covering how others use the information system in their tasks.

Keywords: UML, UML tool, training, learning, teaching, e-teaching, e-learning

Author

Mervi Koivulahti-Ojala
Faculty of Information Technology
University of Jyväskylä
Finland

Supervisors

Lecturer Timo Käkölä
Faculty of Information Technology
University of Jyväskylä
Finland

Professor Emeritus Pertti Järvinen
School of Information Sciences
University of Tampere
Finland

Reviewers

Research Professor Minna Isomursu
Centre for Health and Technology
University of Oulu
Finland

Professor Peter Axel Nielsen
Department of Computer Science
University of Aalborg
Denmark

Opponent

Professor Tommi Mikkonen
Department of Computer Science
University of Helsinki
Finland

ACKNOWLEDGEMENTS

The story of this thesis dates to the beginning of the 1990's when I started working in a department responsible for developing and supporting information systems used by the company's research and development organization. At the time, I was still attending university and completed a master's thesis in 1998. Since then, I have been studying. My motivation has been to deepen my understanding in the area of information systems management. I wish to thank my former and present superiors for their support and encouragement in my efforts to balance learning new things with managing daily tasks, as well as current and former colleagues who believed this thesis would be completed someday.

This thesis and related papers have been written under the supervision of Timo Käkölä. I am greatly indebted to Minna Isomursu and Peter Axel Nielsen for their constructive criticism of the manuscript and their valuable suggestions for improving the text.

I gratefully acknowledge Professor Emeritus Pertti Järvinen for his comments on my study plans, papers, and this manuscript. Discussions in his seminars with students and researchers over the years have been both inspiring and relevant. I also wish to thank my student peers for their valuable feedback over the years. Thanks to Marja-Leena Rantalainen, Sami Kollanus, and Rebekah Rousi who assisted me in finalizing this manuscript.

I want to thank my parents and sisters for giving me the support needed to pursue my interests. Last, but not least, thanks belong to my husband Pasi who has shared both the intellectual interest and the homework, which have made completing this thesis possible. Finally, this thesis is dedicated to my sons Joni and Pauli who have taught me a lot about priorities in life. I love you all.

Kaarina, 14.10.2017
Mervi Koivulahti-Ojala

LIST OF INCLUDED ARTICLES

I Käkölä, T., Koivulahti-Ojala, M. & Liimatainen, J. 2011. An Information Systems Design Product Theory for the Class of Integrated Requirements and Release Management Systems, Journal of Software Maintenance and Evolution: Research and Practice 23 (6), 443-463.

II Koivulahti-Ojala, M. & Käkölä, T. 2009. Framework for Evaluating the Version Management Capabilities of a Class of UML Modeling Tools from the Viewpoint of Multi-site, Multi-partner Product Line Organizations, in Proceedings of the 43rd Hawaii International Conference on Systems Sciences, IEEE Computer Society, Hawaii, USA.

III Koivulahti-Ojala, M. & Käkölä, T. 2012. Design, implementation, and evaluation of a Virtual Meeting Tool-based innovation for UML technology training in global organizations, in Proceedings of the 45rd Hawaii International Conference on Systems Sciences, IEEE Computer Society, Hawaii, USA.

IV Islam, A. K. M. N., Koivulahti-Ojala, M. & Käkölä, T. 2010. A light-weight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool. In Proceedings of the AMCIS 2010.

V Koivulahti-Ojala, M. & Käkölä, T. 2014. Training people to master complex technologies through e-Learning: Case of UML technology training in a global organization, in Proceedings of the AMCIS 2014.

FIGURES

TABLES

CONTENTS

1 INTRODUCTION

Unified Modeling Language™ (UML) is an international standard for systems modeling. UML can be used for specifying, visualizing, and documenting systems. UML was originally developed to provide a unified notation over three object-oriented software development methodologies: the Booch method (Booch, 1994), the object-modeling technique (OMT) developed by Rumbaugh et al. (1991), and object-oriented software engineering (OOSE) developed by Jacobsen et al. (1992). Since 1997, the Object Management Group (OMG) has been developing UML as a standard language and it has gone through several revisions. In its current version UML can be used not only for software systems modeling but also for business process modeling, organization structures modeling, and embedded and real-time systems modeling. In 2005, the UML standard was also published by the International Organization for Standardization (ISO).

The central terms in the UML are models and diagrams. For the purposes of this summary, I adopt the terminology defined by ISO standard (ISO/EIC, 2012) as follows:

- A model captures a view of the physical system, with a certain purpose.
- Diagrams are graphical representations of parts of the UML model. UML diagrams contain graphical elements that represent elements in the UML model.

The UML diagrams represent behavioral and static views of the system. Behavioral diagrams represent the dynamic behavior of the system by visually representing collaboration among the objects or changes to the internal states of the objects. Static diagrams represent structural aspects of the system such as components. The current UML standard version is UML 2.4 and has 14 diagram types: seven diagram types represent static application structure; three represent general types of behavior; and four represent different aspects of interactions. However, according to empirical studies, some of the diagram types are reportedly unused by practitioners (Petre, 2013; Fitsilis et al, 2014). Additionally, practitioners have criticized UML for its complexity, lack of formal semantics, and inconsistency (e.g., Lange et al., 2006; Petre, 2013).

Even though practitioners have raised several concerns regarding UML, an entire industry has emerged to support UML modeling. There are several UML modeling tools (also known as CASE tools) on the market used, according to their vendors, in hundreds of companies for UML modeling. UML modeling tools offer graphical editors to help architects and developers model requirements, architectures, data structures, dynamic behaviors, and other characteristics of systems with UML. Furthermore, these modeling tools can assist with automatization tasks, thus providing opportunities that the UML as a language cannot provide. With the aid of the modeling tools, UML can be used for software code generation and testing. The modeling tools may also be used for reverse engineering, i.e., to generate UML models based on source code. Some modeling tools have a built-in knowledge of UML rules so they can automatically validate the correctness of UML models.

Despite the emerge of an entirely new industry for UML modeling tools there is limited empirical research published in the academic literature regarding using and adopting UML and UML modeling tools; most published studies on UML are individual case studies or surveys. Therefore, no sound estimation for the total number of users using UML or UML modeling tools can be made. The one available literature review related to UML use and adoption is by Budgen et al. (2011) who conducted a systematic literature review to determine how widely the UML's notations and their usefulness had been studied empirically by the end of 2008. They found 49 papers altogether with two rounds of snowballing, of which 11 studies were conducted in industrial settings. Of these 11 studies, only two were related to adopting UML or UML modeling tools. Budgen et al. (2011) concluded there are few field studies and identified UML adoption as a topic needing further investigation.

Despite its 20-year existence, UML adoption is still a relevant research topic. According to a recent study by Fitsilis et al. (2014) about UML usage among IT practitioners in Greece, it was used in nearly half of the projects included in the study. In addition, users declared that they will use UML again in their future projects, and they expect UML usage will increase further during the next several years.

1.1 Research questions

The research objective is providing new knowledge concerning how UML modeling tools are used and evaluated as well as how people are trained to use them. The main research question is: How can a globally distributed product company where UML modeling activities are scattered across different locations and countries implement a UML modeling tool? The answer to the research question was constructed from the five studies.

This thesis' contribution is new knowledge in the IS research community. In particular, the contribution of this summary is the provision of novel insight into: 1) the study process and how the studies related to each other; and 2) the

ways in which new knowledge gained through the studies was used in both the case company and by the IS research community.

All studies were conducted in a global high-technology corporation in the business of developing products in multiple sites with multiple partners. Business units in the company develop one or more products for customers in specific market segments. Thus, product development processes were running simultaneously for several products. A product development process is the process whereby products are developed. It consists of four phases: concept development, system-level design, detailed design, and product testing and refinement (Ulrich, 1995) (Figure 1). Concept development phase activities include selecting technological working principles and choosing functional elements, features, and performance targets to best meet the customer's needs. The system design phase includes developing the product architecture and assigning component development teams to the overall product development team. The detailed design phase is concerned with component design, testing, and production process planning. The product testing and refinement phase involves testing prototypes and implementing any required changes to the component designs.

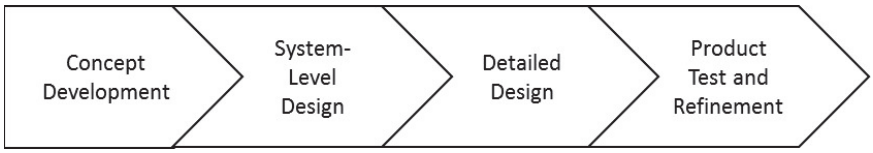

FIGURE 1 Product development process (Ulrich, 1995).

The detailed research questions are described in the following. "Case company" is the term used to reflect that all studies were conducted in the context of one company, and it does not refer to any research approach or method.

1.1.1 Research questions for requirements and release management system (RRMS) (Article I)

In the case company, a new platform organization was established which was responsible for developing common components utilized by different business units in the products they develop. To enhance communication during the requirement and release management process, a new information system was developed. It served all product development phases from concept development (needs, analysis of needs), system level design (features, assignment to teams), detailed design (component and component level release planning) and product test and refinement (releasing of components, test results). The new information system was developed in-house.

Over the years, development of the system has resulted in a complicated information model containing more than twenty different information elements. This study was initiated to better understand which parts of the information system are critical for the success of the system and to develop it further. In sum, the case company's business need was to understand the existing

system's most important properties. The business need was relevant because the product development process was heavily dependent on this information system. The case company considered any system downtime as problematic because it would directly impact sales since no new products could be released without this system. Thus, the information system was considered a critical need by the company. It was used by more than 6000 people in the case company and updated by more than 2000 people in 2008 (Käkölä et al., 2010). By the end of 2008, it contained more than 100 000 active data entities (Käkölä et al., 2010).

The main research question was: "What are the necessary and sufficient properties for an information system which supports an integrated requirement and release management process in a globally distributed product development organization?"

The research sub-questions were:

1) What are the requirements for an integrated requirements and release management system?
2) What is the design for an integrated requirements and release management system?
3) How does the design fulfil the requirements for an integrated requirements and release management system?

The answer to the first research question consists of requirements developed based on the literature and empirical research conducted in the case company. Requirements consist of detailed requirements related to communication, control, change, platform development, and process integration. The answer to the second research question consists of the information model and attributes for the elements presented in the information model. The design was developed based on the empirical research conducted in the case company. The answer to the third research question consists of the results obtained while evaluating the design against the requirements. The design was evaluated against these requirements by analyzing how each requirement was supported by the design.

This study contributed to the main research question by providing new knowledge regarding how UML models were used. According to the study, UML modeling was conducted only when necessary by different tools and diagrams imported or linked to the RRMS. The priorities, schedules, and other information stored in the RRMS were used to focus UML modeling efforts. Thus, this study provided new information about the UML and UML modeling tool usage in the context of the requirement and release management process. Later in the study, it was decided to implement one UML modeling tool globally in the case organization. The UML modeling tool implementation was further studied in Articles II, III, IV, and V.

1.1.2 Research questions for a set of evaluation criteria for UML (Unified Modeling Language) tool version management (Article II)

In the case company, several UML modeling tools were evaluated. During UML modeling tool evaluation, I learned the UML modeling tools did have consider-

able differences regarding version management. Thus, I became interested in identifying the critical features of version management to serve globally distributed product development.

In globally distributed product development modeling activities are scattered across multiple sites and involve multiple teams in different countries. Therefore, proper version management is critical for managing parallel and geographically distributed modeling activities. According to Koivulahti-Ojala and Käkölä (2009), literature does not provide a comprehensive set of evaluation criteria which could be applied in industrial settings to evaluate the version management capabilities of UML modeling tools in the context of globally distributed product development.

The main research question was: "What are the necessary and sufficient properties for version management to support UML modeling in globally distributed product development?"

The research sub-questions were:

1) What is the set of evaluation criteria for version management capabilities of UML modeling tools?
2) How can the set of evaluation criteria be used for evaluating UML modeling tools?

The answer to the first research question is a set of evaluation criteria. The evaluation criteria were developed based on relevant literature and the author's experience while evaluating and adopting a UML modeling tool in the case company.

The answer to the second research question consists of the results of a laboratory test. During laboratory testing, two modeling tools were installed, and features of both tools evaluated and the evaluation results were documented.

This study contributed to the main research question by providing a new set of evaluation criteria which can be used during the evaluation of UML modeling tool version management capabilities to support UML modeling in globally distributed product development.

1.1.3 Research questions for a new training method to support training of UML and UML modeling tool (Article III)

End-user training is complicated to implement in a globally distributed product development company where activities are scattered across multiple sites. During UML modeling tool implementation in the case company, a survey was implemented to verify user satisfaction with the tool and the service (Article IV). Based on the survey results, the support team decided to implement a new training method. To be successful, the training should positively impact learners' skills, knowledge, and motivation and learners should be satisfied with the training. A virtual meeting tool (VMT) was chosen to support delivery of training because it was already in use for meetings.

The research question was: Can the UML and UML modeling tool training be organized and delivered through a VMT so that learners are satisfied with

the training and the training positively impacts their skills, knowledge, and motivation?

The research sub-questions were:

1) What is the design for a UML modeling language and UML modeling tool training organized and delivered through a virtual meeting tool?
2) How does the design fulfil the target of learners' satisfaction with the training and the training positively impacting their skills, knowledge, and motivation?

The answer to the first research sub-question is a method as described by March and Smith (1995). The training method was described in terms of content, organization of training, training materials, and trainers' skills and knowledge. The training method was described based on the case study conducted in the case company.

The answer to the second research sub-question consists of case study results. According to the study, users' skills, knowledge, and motivation were improved and learners were satisfied with the training organized through a VMT (Koivulahti-Ojala and Käkölä, 2012).

This study contributed to the main research question by providing new knowledge regarding how UML and UML modeling tool training can be organized through a VMT.

1.1.4 Research question for a new measurement instrument (Article IV)

There was a target to routinely measure both user satisfaction and service quality for the tools utilized by users in the case company. It was most appropriate to measure the user satisfaction and service quality variables after the UML modeling tool was adopted. There were three requirements for a new measurement instrument: 1) it should measure both the service quality and the user satisfaction with respect to the tool; 2) there should be no more than 10 questions (including two standard questions of location and frequency of usage), 3) the instrument should be applicable to further develop the service and the tool. The IS research community has delivered many comprehensive instruments to measure user satisfaction and service quality (e.g., Petter et al., 2008). However, the first and the second requirement limited the choice of instrument to an existing instrument because there was none available to cover both the service quality and the tool related satisfaction while utilizing only eight questions.

The research question was: "Is it possible to create a new adequately reliable and valid measurement instrument with eight items to measure both user satisfaction and service quality?"

The main contribution of this research was the design, implementation, and evaluation of a new 8-item instrument to evaluate users' satisfaction with the tool and the services supporting its use. The results detained from analyzing the two surveys conducted in the case company to measure user and service satisfaction with a UML modeling tool indicated the instrument has adequate reliability and validity. Furthermore, it was used successfully to improve the service. It is short, easy to use, and appropriate for both practical and research purposes.

During the study, two rounds of surveys were conducted amongst users in the case company. When the results of the first survey were analyzed, it was concluded that users are not satisfied with the existing training and the decision was made to implement a new training method. This new training method was the subject of the study in Article III.

This study contributed to the main research question by providing new knowledge regarding how to evaluate users' satisfaction with the UML tool and the services supporting the tool.

1.1.5 Research question for a longitudinal study about UML and UML modeling tool training (Article V)

E-teaching tools facilitate asynchronous (e.g., Wikis) and synchronous (e.g., video-conferencing) learning. In the case company, different e-teaching tools had been used routinely during meetings for several years. Later on, these tools were deployed for UML and UML modeling tool training. The case company was interested in understanding which e-teaching tools were most applicable to UML and UML modeling tool training. It was especially unclear whether e-teaching tools can be deployed to teach the following categories of knowledge: application, business context, and collaborative task knowledge. Application knowledge covers commands and tools embedded in the information system; business context knowledge covers the use of information systems to effectively perform business tasks; and, collaborative task knowledge covers how others use the information system in their tasks.

The research question was: Which classes of e-teaching tools are most applicable for organizing and delivering technology training allowing large numbers of learners to become trained in application, business context, and collaborative task knowledge needed to master the UML and UML modeling tool?

According to this longitudinal case study, an intranet and a virtual meeting tool (VMT) were used to support UML and UML modeling tool training in application knowledge, business context knowledge, and collaborative task knowledge. In addition, Wikis, discussion forums and e-mail were used to support UML and UML modeling tool training but not for all three types of knowledge.

This study contributed to the main research question by providing new knowledge concerning which tools have been used to support delivering of UML and UML modeling tool training in a globally distributed product company.

The five articles are interrelated (Figure 2). Article I provided insight into how UML modeling tools were used during the product development process. Furthermore, for the case company, it provided new information that UML diagrams are created but employees are using different UML modeling tools to do so. Thus, it established a common need for the UML modeling tool implementation. UML modeling tool evaluation, use, and training was discussed in Articles II, III, IV and V.

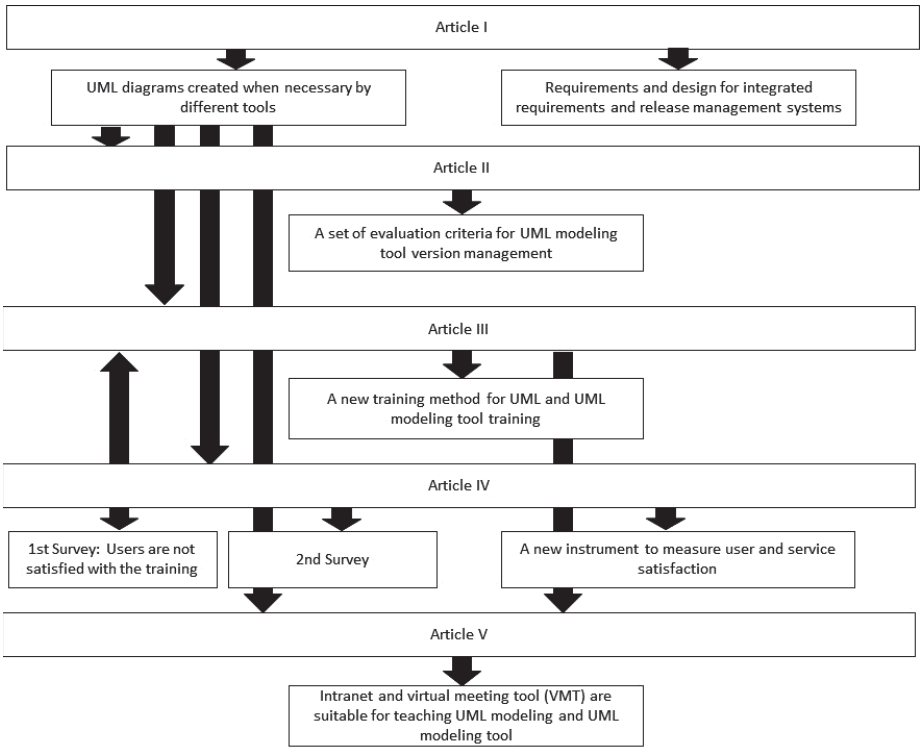

FIGURE 2 Relationships between the articles.

In Article II the evaluation criteria for UML modeling tool version management in the context of globally distributed product development were introduced. According to my experience, UML modeling tools did differ in terms of features' capability of supporting globally distributed product development. According to Koivulahti-Ojala and Käkölä (2009), literature does not provide a comprehensive set of evaluation criteria which could be applied in industrial settings to evaluate the version management capabilities of UML modeling tools in the context of globally distributed product development. Thus, both research and practice were triggers for this study.

When a new UML modeling tool was implemented in the case company, it was relevant to measure users' satisfaction with the tool and the service (Article IV). This article provided new information for the research community related to how users' satisfaction with the tool and service can be measured. Moreover, when the first survey's results were analyzed in the case company, it was concluded the users were not satisfied with the training and a new training method using a virtual meeting tool (VMT) was implemented. This new training method was the subject of Article III's study. Finally, Article V provided new information as a longitudinal study on how the new training method through VMT among other training methods evolved over time. Thus, Article V extended the study in Article III in two respects: several e-teaching tools were studied instead of one and the study was longitudinal.

1.2 Authors' contribution to the included articles

This thesis consists of one journal article and four conference articles presented in the following:

I Käkölä, T., Koivulahti-Ojala, M. & Liimatainen, J. 2011. An Information Systems Design Product Theory for the Class of Integrated Requirements and Release Management Systems

II Koivulahti-Ojala, M. & Käkölä, T. 2009. Framework for Evaluating the Version Management Capabilities of a Class of UML Modeling Tools from the Viewpoint of Multi-site, Multi-partner Product Line Organizations

III Koivulahti-Ojala, M. & Käkölä, T. 2012. Design, implementation, and evaluation of a Virtual Meeting Tool-based innovation for UML technology training in global organizations

IV Islam, A. K. M. N., Koivulahti-Ojala, M. & Käkölä, T. 2010. A light-weight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool

V Koivulahti-Ojala, M. & Käkölä, T. 2014. Training people to master complex technologies through e-Learning: Case of UML technology training in a global organization

I was the main author of Articles II, III, and V. I was responsible for research planning, writing the article, conducting the literature review, creating the evaluation framework, and analyzing the modeling tools for Article II. The second author contributed to writing and evaluation framework creation. I was responsible for research planning, writing the article, conducting the literature review, and conducting and analyzing interviews for Article III. The second author contributed to writing and analyzing interviews. I was responsible for research planning, writing the article, and collecting and analyzing data for Article V. The second author contributed to writing and analyzing research results. I contributed to research planning, writing, requirements and design creation and evaluation, and analyzing interviews in Article I. I contributed to research planning and writing, as well as implementing and developing the instrument and interpreting the statistical analysis results in Article IV.

1.3 Structure of the thesis

This thesis is organized as follows. In Chapter 1, I present the research objective and the research questions. In Chapter 2, I describe the five articles included in this thesis, related research methods, the case company and my role in the case

company. In Chapter 3, I evaluate the relevance and rigor of each study in three respects. First, I evaluate the relevance of each study from the perspective of the case company after the study was conducted. Then, I evaluate the relevance of each study based on how results were used in IS research. Finally, I evaluate the rigor of each study. In Chapter 4, I summarize contribution of each study towards answering the main research question. In Chapter 5, I summarize theoretical and practical contributions, limitations, and the implications for further research.

2 SUMMARY OF ARTICLES

In this chapter, I present the five articles, the research approach, research methods applied in these articles, and the case company, as well as describe my role in the case company. The term "case company" is used to reflect all studies were conducted in the context of one company. The term "case company" does not refer to any research approach or method. The description of each article contains the research problem of the article, the research approach and process, the results that I emphasize in the article, limitations of the research, and contribution to this thesis.

Each study's research question originated from work experience and their research approaches and methods were chosen prior to beginning the research process for each study. Therefore, the research approach and methods are described in the context of each article. My studies represent a mathematical research approach, research approach where is studied what is reality, and research stressing utility of innovations (Järvinen, 2012, p. 10). Järvinen (2012, p. 10) introduced a taxonomy to help select the most suitable research approach for a research problem (Figure 3). He first differentiates research stressing what is reality from mathematical research approaches. In mathematical studies, a certain theorem, lemma, or assertion is proved. He further distinguishes research stressing what is reality and research approaches stressing utility of innovations. Research approaches studying utility of innovations he further distinguishes into innovation building approaches and innovation evaluation approaches. He divides the research approaches studying what is reality into conceptual analytical approaches (i.e. research methods for theoretical development) and empirical research approaches.

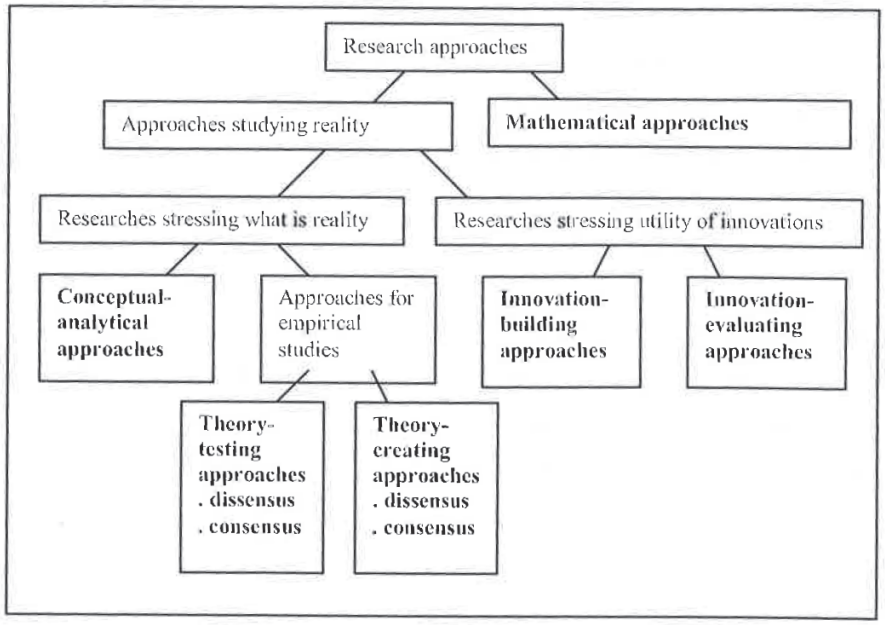

FIGURE 3 Research approaches (Järvinen, 2012, p. 10).

Järvinen (2012) distinguishes research approach and research methods. Research methods may support different research approaches. Theory-creating research approaches include case studies (Yin, 1989). Additionally, Lee (1989) presented a specific version of the case study which should be classified as a theory-testing approach. Thus, case research methods may be applied in the context of different research approaches. In conceptual-analytical research, basic assumptions behind constructs are analyzed; theories, models and frameworks used in previous studies are identified, and logical reasoning is thereafter applied; then, a new tentative theory, model or framework describing a certain part of the reality is developed. Theory-testing research methods include laboratory experiments, surveys, field studies, and field tests.

My studies represent different research approaches (Table 1). Article I represents a research approach where the utility of innovation is studied, i.e., the utility of a requirements and release management system (RRMS) is analyzed. We followed a design method developed by Walls et al. (1992). In Article II, new evaluation criteria are conceptual-theoretically derived from identified user needs in the context of globally distributed product development and measured in natural settings. Article III describes the design and re-design of training arrangements, and therefore represents a longitudinal design project. In Article IV, a new measurement instrument was created for a reflective construct that could not be measured as such, following instructions presented by Churchill (1979). This study represents a mathematical research approach. In Article V, a tentative list of the e-learning tools most suitable for UML modeling

and UML modeling tool training is created, thereby representing a theory creation approach. This study is a longitudinal case study.

TABLE 1 Research approaches and research methods applied in each study.

Article	Research question	Research approaches mapped according to taxonomy by Järvinen (2012, p. 10)	Research method
Article I	What are the meta-requirements and meta-design of Information System Design Theory for the class of integrated requirements and release management system (RRMS) in a globally distributed product development organization?	Innovation building and innovation evaluation	Design method developed by Walls et al. (1992)
Article II	What are the necessary and sufficient properties for UML modeling tool version management to support UML modeling in globally distributed product development?	Innovation evaluation	A controlled test to evaluate the set of evaluation criteria in a laboratory test
Article III	Can the UML and UML modeling tool training be organized and delivered through a VMT so that learners are satisfied with the training and the training positively impacts their skills, knowledge, and motivation?	Innovation building and innovation evaluation	Design method developed by Peffers et al. (2007)
Article IV	Is it possible to create a new adequately reliable and valid measurement instrument with eight items to measure both user satisfaction and service quality?	Mathematical approach	Churchill (1979)
Article V	Which classes of e-teaching tools are most applicable for organizing and delivering technology training allowing large numbers of learners to become trained in application, business context, and collaborative task knowledge needed to master the UML modeling language and UML modeling tool?	Theory creating	Case study Yin, 2003

2.1 An Information Systems Design Product Theory for the Class of Integrated Requirements and Release Management Systems (Article I)

2.1.1 Research problem and research strategy

Distributed product development organizations need to collect, analyze, and utilize requirements. Well-defined requirements are prerequisites for assigning appropriately scoped projects to internal units and service providers for implementation (Carmel and Agarwal, 2002; Adelson and Soloway, 1985). Release management is concerned with the identification, packaging, and delivery of product's elements (ISO/IEC TR 19759, 2010). Salo and Käkölä (2005) developed an Information System Design Theory (ISDT) for the class of Requirements Management Systems (RMS) to help IT practitioners build RMS for creating, prioritizing, and storing requirements. Methodologically, their work was based on Walls et al. (1992) who articulated how to construct and test an ISDT capable of guiding the design of a specific class of information systems. They argue that the information systems "field has now matured to the point where there is a need for theory development based on paradigms endogenous to the area itself" and call for Information System Design Theories to fill this need. An ISDT is "a prescriptive theory based on theoretical underpinnings which says how a design process can be carried out in a way which is both effective and feasible". According to Salo and Käkölä (2005), the RMS's benefits were limited when the instances were not integrated with the systems used in the downstream phases to provide transparent end-to-end support throughout the product development. For example, customer representatives responsible for entering requirements were not able to use the RMS instances to follow-up concerning when the requirements would be implemented. The Information System Design Theory's scope should thus be broadened to design systems supporting the whole life cycle more comprehensively.

In this study, knowledge in the case company and existing research knowledge were used to create a new Information System Design Theory. According to Walls et al. (1992) ISDT prescribes both the design product and process aspects of a class of IS, that is, what are (1) the value propositions to be fulfilled by implementing an instance of the class, (2) meta-requirements describing the problem(s) to be solved by the class, (3) the meta-design prescribing the solution for the problem(s), and (4) applicable kernel theories from social and natural sciences for understanding and/or solving the problem(s) shared across all products within the class, and how the products should be built. In this study, the research focused on meta-requirements and meta-design. Therefore, in terms of Information System Design Theory, the research question was as follows:

- What are the meta-requirements and meta-design of Information System Design Theory for the class of an integrated requirements

and release management system (RRMS) in a globally distributed product development organization?

And the research sub-questions were:

1) What are the meta-requirements for the class of integrated requirements and release management system (RRMS)?
2) What is the meta-design for the class of integrated requirements and release management system (RRMS)?
3) How does the meta-design fulfil the meta-requirements for the class of an integrated requirements and release management system (RRMS)?

2.1.2 Research process

The research began by conducting a literature review to develop preliminary meta-requirements. Afterwards, interviews and other data gathering methods were used to gain a comprehensive understanding of RRMS features and usage in the case company. Two of the paper's authors had access to all relevant information and could interact with people who were involved with the RRMS design. We observed use of the RRMS, analyzed documentation, held informal discussions with various stakeholders, and conducted six semi-structured interviews with middle-level managers who were involved with the design and use of the RRMS for process improvement. Based on the interviews, meta-requirements were modified and the information model and attributes for the elements created. Finally, the design was evaluated against the meta-requirements by analyzing how each meta-requirement is supported by the meta-design. The evaluation was documented as part of the paper.

I contributed to this study through research planning, writing, meta-requirements and meta-design creation and evaluation, and planning and analyzing interviews. Because I was working as a System Specialist in the case company and responsible for this information system, we decided interviews should be conducted by another research team member.

2.1.3 Research results and contribution to this thesis

The main contributions of the article were the meta-requirements of the Information System Design Theory and a meta-design that meets the meta-requirements. Meta-requirements were developed based on the literature and empirical research conducted in the case company. Meta-requirements consist of fifteen requirements related to communication, control, change, platform development, and process integration. Meta-design consists of an information model and attributes for the elements presented in the information model. In their work, Salo and Käkölä (2005) did not propose an information model. Their approach was based on a two-level document structure. In our work, a new information model was proposed (Figure 4). The meta-design was validated against meta-requirements by analyzing how each meta-requirement was supported by the meta-design.

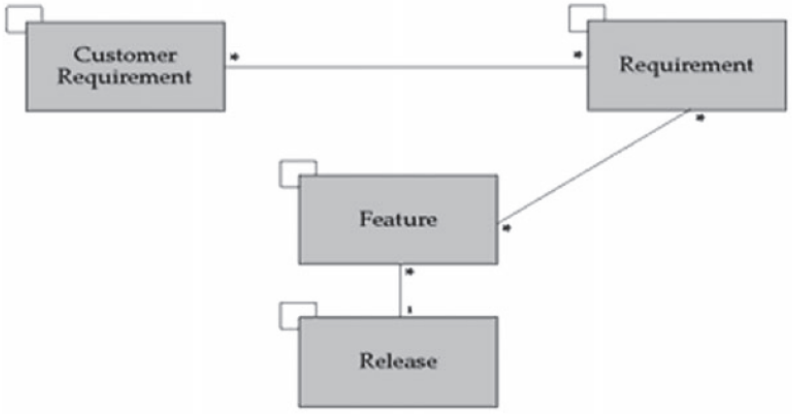

FIGURE 4 Information model of the meta-design of the design product theory for the class of RRMS (Käkölä et al., 2011).

Research results are relevant for IS researchers whose research interests are related to requirement or release management process development, information systems supporting the requirement or release management process, process integration, or integration of information systems. Information System Design Theory for the class of RRMS facilitates theory development in the context of RRMS. This may include, but is not limited to, theory development in the context of 1) requirements management systems, 2) release management systems, 3) integrated requirements and release management systems (RRMS), and 4) integration of requirements and release managements processes.

Research results are relevant for R&D managers who can take advantage of the results when planning, evaluating, implementing, or deploying information systems to support requirement and release management. IT practitioners benefit from the results when planning, implementing, or evaluating such systems.

This study contributes to this thesis by providing information about UML and UML modeling tool usage in the context of the requirement and release management process. According to the study, if requirements in the RRMS needed specific product UML models to make them understandable to executives, managers, service providers, or other critical stakeholders, the models were crafted in appropriate modeling environments as necessary and hyperlinked or, sometimes, imported into the RRMS. The priorities, schedules, and other information stored in RRMS were used to focus UML modeling efforts. UML models where thus created but they did not cover the whole system. This result complements Nugroho and Chaudron's (2008) work. According to their survey of 80 software professionals who use UML, they found that most of the UML models were not complete, meaning they did not cover all elements of the system. Results of our study are thus in line with their study.

2.1.4 Limitations and future research

The main limitation of this research is the theory's design product effectiveness hypotheses (clarifying the expected organizational benefits from using an RRMS instance derived from the class of RRMS) are missing. The hypotheses are needed for the empirical validation and possible revision of the theory in future research. From a methodical perspective, the Information System Design Theory is applied here regarding Walls et al.'s (2004) "level 2 use" where Information System Design Theory is used as a common language and framework for determining the meta-requirements. Walls et al. (2004) propose the application of Information System Design Theory is more advanced when kernel theories are used to create new insights or even further to propose advancements for Information System Design Theory.

One limitation of this study is the meta-requirements and meta-design were created in the context of one company and therefore may not be suitable for other companies' use. Because the case company has successfully used the application for several years with different products, inter-organizational setups, and partners, and due to substantial effort made during the research process to study the literature we were, however, confident RRMS is also suitable for purposes other than in the case company.

In the original paper, the research method was described as a case study. Case study in the original paper reflected the data gathering methods.

2.2 A Framework for Evaluating the Version Management Capabilities of a Class of UML Modeling Tools from the Viewpoint of Multi-site, Multi-partner Product Line Organizations (Article II)

2.2.1 Research problem and research strategy

Unified Modeling Language™ (UML) is used in globally distributed product development organizations for modeling the architecture, detailed design, and automation of software code generation and testing. In such organizations, modeling activities are typically scattered across multiple sites and involve multiple teams in different countries. UML modeling tools utilizing version management are critical for managing parallel and geographically distributed modeling activities. According to (Koivulahti-Ojala and Käkölä, 2010), the literature does not provide a comprehensive set of evaluation criteria which could be applied in industrial settings to evaluate the version management capabilities of UML modeling tools.

In the second research paper we built and evaluated a framework for evaluating the version management capabilities of UML modeling tools. The research problem of the second research paper was:

- What are the necessary and sufficient properties for version management to support UML modeling in globally distributed product development?

The sub-research questions were:

1) What is the set of evaluation criteria for version management capabilities of UML modeling tools?
2) How can the set of evaluation criteria be used for evaluating UML modeling tools?

This research represented the evaluation phase of design science research (Peffers et al., 2007). Peffers et al. (2007) proposed a six-phase method based on Nunamaker et al. (1990), Walls et al. (1992, 2004), Hevner et al. (2004), March and Smith (1995), and Rossi and Sein (2003) in the design research discipline. Their method consists of problem identification and motivation, definition of the objectives for a solution, design and development, demonstration, evaluation, and communication. This research represents evaluation phase of the design science research, i.e., evaluation of existing UML modeling tools. The research approach was conceptual-analytical in the sense a new set of evaluation criteria was created based on an analysis of existing research in SW version management and documented requirements for assets needing to be managed in product line organization based on the author's experience during the evaluation and adoption of a UML modeling tool in the case company (research sub-question 1). However, a controlled test approach was applied when the set of evaluation criteria was evaluated in a laboratory test (research sub-question 2). During laboratory testing, I was responsible for installing the version management tool, two UML modeling tools (client and servers), analyzing their features against the set of evaluation criteria, and reporting of the results.

2.2.2 Research process

The set of evaluation criteria was developed based on the literature and experiences reported in the paper. Evaluation criteria were applied in laboratory tests for testing two UML modeling tools. Laboratory tests included installing the tools, analyzing each feature, and documenting the results. According to the results, it was possible to differentiate these two UML modeling tools according to their version management capabilities. The case company representatives used and were more favorable for the version management capabilities of the UML modeling tool that, according to laboratory testing, got better scores. This indicated the tool receiving the highest scores is likely to be more capable of supporting version management of UML models in a globally distributed product development organization.

In this study, I was responsible for research planning and writing the article, as well as evaluation framework creation and analyzing the modeling tools. The second author contributed to writing and evaluation framework creation.

2.2.3 Research results and contribution to this thesis

The main contribution of this research was the set of validated evaluation criteria for the version management capabilities of UML modeling tools. The laboratory tests indicated the set of evaluation criteria is feasible for laboratory testing, which means it can be applied by organizations to evaluate the version management capabilities, and the UML modeling tool receiving the highest scores is more likely to meet the requirements of a distributed product development organization.

This study contributes to this thesis by providing new knowledge about UML modeling tool evaluation. Research results are relevant to IS researchers whose research interests are related to UML modeling tools. Research topics may include, but is not limited to, theory development in the context of UML modeling tool evaluation.

Research results are relevant to R&D management and IT practitioners who can take advantage of the results when evaluating the UML modeling tool's version management capabilities. Global R&D organizations evaluating a UML modeling tool benefit from the framework as they can use it during the evaluation or the evaluation results of these two modeling tools. This is highly beneficial as it requires substantial effort to install the tools as well as complete the evaluation, especially for medium size companies. The total effort required for both installation and evaluation was several man-months.

2.2.4 Limitations and future research

The main limitation of this study is the evaluation was conducted only in laboratory settings. More comprehensive results could possibly be obtained by extending testing from the laboratory to case organizations where, for example, successful deployment of UML modeling tool version management capabilities can be evaluated. Furthermore, a set of evaluation criteria can be extended to include other features of UML modeling tools. In this article, a generic UML modeling tool feature list was proposed and by creating a set of evaluation criteria for all features relevant to the research target group could be extended to, for example, IT practitioners when evaluating and deploying UML modeling tools.

2.3 Design, implementation, and evaluation of a Virtual Meeting Tool-based innovation for UML technology training in global organizations (Article III)

2.3.1 Research problem

End-user training is complicated to implement in globally distributed product development organizations where its business activities are scattered across

multiple sites. Virtual meeting tools (VMT) enable synchronous communication globally through interactive audio, online chats, video, and sharing presentations. VMT provides a potentially cost-effective way to deliver training in even complex topics to large numbers of people in global settings.

In this paper, a case study was conducted in a globally distributed R&D company to describe the design, implementation, and evaluation of a method for teaching skills and knowledge needed to use UML and UML modeling tool. The UML was used in the case company for modeling architecture and detailed design. The UML modeling tool was used to create and maintain models and diagrams, the version management of models, and reverse engineering of code. In the case company, the target was learners' satisfaction with the training and it should positively impact learners' skills, knowledge, and motivation regarding application, business context, and collaborative tasks.

The research problem of the third research paper was:

- Can the UML and UML modeling tool training be organized and delivered through a VMT so that learners are satisfied with the training and the training positively impacts their skills, knowledge, and motivation?

The sub-research questions were:

1) What is the design for a UML and UML modeling tool training organized and delivered through a virtual meeting tool?
2) How does the design fulfil the target of learners' satisfaction with the training and the training positively impacting their skills, knowledge, and motivation?

The answer to the first sub-research question is a method described by March and Smith (1995). The training method was described regarding content, organization of training, training materials, and trainers' skills and knowledge. The training method was depicted based on a case study conducted in a case company. The answer to the second sub-research question consists of evaluation results obtained from the case company. According to the study, users' skills, knowledge, and motivation were improved and learners were satisfied with the training organized through VMT (Koivulahti-Ojala and Käkölä, 2012). The research approach was constructive considering a new training method was built and evaluated. Evaluation was based on interviews to verify the impacts on learners' skills, knowledge, and motivation, and perceived learner satisfaction with the new training method. Evaluation of the training method was conducted from the individual learner's perspective.

2.3.2 Research process

To answer the research questions, the six-phased design research methodology presented by Peffers et al. (2007) was utilized and four of the six steps were focused on during the research process. First, *Problem Identification and Motivation* revealed the UML training-related research did not provide any insights into the design and implementation of VMT innovations for UML training. A detailed literature review is included in the paper. Second, *Objectives for an Innova-*

tion were defined to resolve the problem based on the case company's experiences. The objective for the research was: "Can the UML modeling and modeling tool training be organized and delivered through a virtual meeting tool in ways that learners are satisfied with the training and the training positively impacts their skills, knowledge and motivations?". Third, the new training method's key components such as content, organization of training, training materials, and trainers' skills and knowledge were *Designed and Developed*. Fourth, learner satisfaction and improvements in skills, knowledge, and motivation were *Evaluated*. Evaluation was based on interviews to verify the innovation's impacts on skills, knowledge, and motivation and perceived learner satisfaction regarding the new training method. Based on results from the interviews, the training method proved successful in improving skills, knowledge, and motivation in the case company and learners were satisfied with it. As the method was already in use when the study was started, instead of the demonstration phase it was focused on evaluating the existing method. Therefore, the demonstration phase was not documented. The final phase of communication was not possible to complete fully because at the time of the study, the case company was not willing to divulge any information related to the training costs.

In this study, I was responsible for research planning, writing of the article, and conducting and analyzing interviews. The second author contributed to writing and analysis of interviews.

2.3.3 Research results and contribution to this thesis

The main contribution of this research was the design, implementation, and evaluation of a VMT-based training method for teaching the UML and UML modeling tool. Design and implementation of training was specified in terms of content, organization of training, training materials, and trainers' skills and knowledge. Based on the evaluation, VMT was applied in the case company for UML training successfully in terms of learner satisfaction, and improved skills, knowledge and motivation.

IT practitioners benefit from the new training method when planning, implementing, and evaluating the UML and UML modeling tool training. R&D management can take advantage of the results when planning, implementing, and evaluating the UML and UML modeling tool training. IT practitioners and R&D management can take advantage of the results when making decisions about VMT usage in UML and UML modeling tool training.

This study contributes to this thesis by providing new knowledge about UML and UML modeling tool teaching. Research results are relevant for IS researchers whose research interests are related to UML modeling, UML modeling tools, or end-user training. Research topics may include, but is not limited to, new method development and evaluation in the context of UML modeling language training, UML modeling tool training, and VMT-based training.

2.3.4 Limitations and future research

Subjective opinions of interviewees do not necessarily correlate with real improvements in skills and knowledge or learner satisfaction. However, other data sources within the company support the interview results. First, a user satisfaction survey completed in the company indicated after the UML and UML modeling tool training sessions were provided, user satisfaction was increased (see details in Article III). Second, the case company attempted other ways of supporting end-users' efforts to learn UML technology but they were unsuccessful in terms of popularity amongst the end-users.

In the original paper, the research method was described as a case study. Case study in the original paper reflects the data gathering methods.

2.4 A lightweight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool (Article IV)

2.4.1 Research problem and research strategy

Existing research in information systems evaluation considers user satisfaction and service quality as central constructs and has produced comprehensive approaches using multidimensional instruments (DeLone and McLean,1992; Delone, 2003; Doll and Torkzadeh, 1988, 1991; Ives et al., 1983; Petter et al., 2008; Pitt et al. 1995; Smithson and Hirschheim, 1998; Symons, 1991). From a distributed product development organization's viewpoint, there are two main limitations in the current research. First, based on experiences in the case company, when collecting data with several surveys or using each with a large set of questions, response rate was low. Secondly, the case company's representatives were not satisfied with the current IS ZOT SERVQUAL instrument as the users may not be able to meet the support personnel face-to-face to evaluate physical facilities, equipment, or personnel-related tangibles and therefore cannot reliably answer the related questions. IS ZOT SERVQUAL (Kettinger and Lee, 2005) deploys 54 questions to be answered for IS service quality.

In the case company, there was a target to periodically measure both user satisfaction and service quality for the tools used. This was a mandatory action because the case company was committed to fulfil the criteria set in ISO 9000 certification to maintain customer satisfaction. Some existing instruments were presented (for SERVQUAL, Kettinger and Lee, 2005; Jiang et al., 2002; Pitt et al., 1995, for UIS Ives et al., 1983, for EUCS, Doll and Torkzadeh, 1988), but according to the case company representatives, they were not suitable for its purposes. Therefore, requirements were set for the new measurement instruments as follows: 1) it should measure both the service quality and user satisfaction regarding the UML tool; 2) there should be no more than 10 questions (including two

standard questions of location and usage frequency), 3) the instrument should be applicable to further develop the service and the tool.

In the fourth research paper, we created and evaluated a lightweight 8-item instrument to measure user satisfaction and the quality of service experienced by the users of a UML modeling tool in the case company. The research problem of the fourth research paper was:

- Is it possible to create a new adequately reliable and valid measurement instrument with eight items to measure both user satisfaction and service quality?

The research approach was constructive considering a new instrument was created and evaluated. However, a theory testing approach was applied when the instrument was evaluated.

2.4.2 Research process

The research process followed instructions presented by Churchill (1979) focusing not on individual measures but the overall validity of the new measurement instrument:

1. Specify the domain of the construct.
Domain of the construct was specified as user satisfaction measurement and service quality measurement for an information system.

2. Generate a sample of items
A sample of items was generated based on existing measurement instruments and reviewed by case company representatives.

3. Collect the data
Data was collected via survey.

4. Purify measure
Instead of purification, this phase's focus was on central tendency computation, regression analysis, item to criterion correlation, and item to total correlation. Item to total correlation was analyzed to ensure higher model reliability. As a part of this analysis it was noticed that overall satisfaction was not explained by Item 6 (How satisfied are you with training available?) featured in the first survey. However, the survey item was not removed or changed as the users were, according to the first survey's results, not satisfied with the training. Based on the first survey conducted in the case company, it was concluded the communication and training practices had to be improved because the means of questions related to instructions, user guides, and training were lower than the mean of all questions. Information sharing with the users was improved in several ways and a new training method was implemented.

5. Collect the data
Data were collected via survey.

6. Assess reliability with new data.
To ensure higher model reliability, the correlation of each item's score with the total of all items' scores was computed. A threshold of 0.45 was used for this validity check. The correlation values were well above the threshold except the result of Q6 in the 1st survey (see the explanation in Purify measure). Therefore, we concluded the instrument has adequate reliability.

7. Assess construct validity.
To ensure statistical conclusion validity (Straub et al., 2004), we performed a regression analysis. According to the results, all the questions had at least a modest fit when following rules described by Bryman & Cramer (1999).

In this study, I contributed to research planning, writing, implementation, and deployment of the instrument, as well as statistical analysis and interpreting statistical analysis results.

2.4.3 Research results and contribution to this thesis

The main contribution of this research was the design, implementation, and evaluation of a new measurement instrument to evaluate users' satisfaction with the UML tool and services supporting tool. The list of items in the instrument is given in the Appendix of Article IV.

From an IS researcher's viewpoint, an analysis of the results of two surveys conducted in a case company indicates the new measurement instrument had adequate reliability and validity. Compared to the available existing instruments, the number of questions was significantly smaller and therefore provides IT practitioners with a new measurement instrument that can be applied instead of those currently used for similar purposes, specifically to evaluate satisfaction and service quality experienced by users of an information system. Furthermore, in the case company it was proven easy to use and appropriate for further improving the service and tool.

IT practitioners benefit from the proposed instrument when measuring user satisfaction and service quality for information systems. IS researchers can benefit from the results including, but not limited to, theory development in the context of information systems-related service quality measurement and user satisfaction measurement. Research results are thus relevant for IS researchers whose research interests are related to measurement of user satisfaction or service quality for information systems.

This study contributes to this thesis by providing new knowledge about user satisfaction and service quality experienced by users of the UML modeling tool. According to the results, by providing new training methods and improving communication, overall satisfaction with the UML modeling tool and related services were improved in the case company.

2.4.4 Limitations and future research

The main limitation of this study is the new measurement instrument was tested only in the context of one application and organization. Future research is needed to validate the instrument in the context of other organizations and other classes of information systems. However, the instrument was designed to be generally applicable for evaluating a variety of systems and services.

2.5 Training people to master complex technologies through e-Learning: A case study of UML technology training in a global organization (Article V)

2.5.1 Research problem

E-teaching tools facilitate asynchronous and synchronous collaboration. Examples of e-teaching tools include Wikis, intranet, internet, e-mail, discussion forums, and virtual meeting tools (VMT) which enable real-time interactions through features such as chat tools, audio, video, and user interfaces for screen sharing. E-teaching tools may provide a cost-effective way to train many people simultaneously in global settings to leverage complex technologies such as the UML modeling language and UML modeling tool. We consider the UML modeling language and UML modeling tool training a complex technology to teach for the following reasons. First, there is a high number of diagrams and symbols with which learners should become familiar. Second, modeling requires both the understanding of UML and the ability to use the UML modeling tool. Third, using UML requires long-term training and learning efforts (Dori, 2002; Kobryn, 2002).

According to a literature review conducted by Koivulahti-Ojala and Käkölä (2014), there are no longitudinal studies concerning UML or UML modeling tool training through e-teaching tools in industrial settings. It is unclear whether e-teaching tools can be deployed to learn three types of knowledge needed as a user in the context of applications supporting collaboration: application, business context, and collaborative task knowledge (Kang and Santhanam, 2003). Kang and Santhanam (2003) identified three knowledge domains user training should deliver in the context of information systems supporting collaboration: application knowledge covering commands and tools embedded in the information system; business context knowledge covering the use of information systems to effectively perform business tasks; and collaborative task knowledge covering how others use the information system in their tasks.

In this paper, a longitudinal case study was conducted in a globally distributed R&D organization to evaluate several classes of e-teaching tools utilized in supporting the teaching of application, business context, and collaborative task knowledge required for UML modeling language and UML modeling

tool implementation. The UML modeling language was used in the case company for modeling architecture and detailed design. The UML modeling tool was used to create and maintain models, to create and maintain diagrams, version management of models, and reverse engineering of code. The research problem of the fifth research paper was:

- Which classes of e-teaching tools are most applicable for organizing and delivering technology training allowing large numbers of learners to become trained in application, business context, and collaborative task knowledge needed to master the UML modeling language and UML modeling tool?

E-teaching tools included Wikis, intranet, e-mail, discussion forum and virtual meeting tools (VMT) which enable real-time interactions through features such as chat tools, audio, video, and user interfaces for screen sharing.

2.5.2 Research strategy and process

In case studies, IS researchers find out conditions in the target organization by making observations, interviewing, archiving, and recording (Yin, 2003, p. 83). Benbasat et al. (1987) state three reasons why case studies are suitable for IS research:

- The IS researcher can study the information system in a natural setting.
- The IS researcher can answer "how" and "why" questions.
- Case study is an appropriate way to research an area in which few previous studies have been carried out.

All three reasons were valid in this research. According to the literature review conducted and reported as a part of the paper, the extant research on e-teaching for UML modeling language and UML modeling tool training consists of a few papers covering only a few e-teaching tools. Longitudinal studies are missing. Therefore, conducting a study in a natural setting brings new information about the usage of several e-teaching tools as well as about long-term usage of e-teaching tools.

During the case study, each e-teaching tool used for UML and UML modeling tool training were listed, and the content of the training analyzed and mapped to the application, business context, and collaborative task knowledge (Kang and Santhanam, 2003). To provide a longitudinal view, each e-teaching tool's usage was described in detail during the years 2008, 2010, and 2013, and finally, an overview of each e-teaching tool's usage over time was created. Sources for the information were interviews, documents, meeting memos, intranet and e-mails.

Evaluating the success of e-teaching tools in the case company was conducted based on the usage of the tools. In 2013, intranet, email, and VMT were the only e-teaching tools in use. Additionally, VMT tool usage in on-line training sessions covering the UML modeling language and UML modeling tool was evaluated as a separate study (Koivulahti-Ojala and Käkölä, 2012). According to

the study, skills, knowledge, and motivation of users were improved and learners were satisfied with the training (Koivulahti-Ojala and Käkölä, 2012).

In this study, I was responsible for research planning, writing the article, and collecting and analyzing data. The second author contributed to writing and analyzing research results.

2.5.3 Research results and contribution to this thesis

Based on the case study, face-to-face training and support were accompanied by a wide variety of e-teaching tools including Wikis, discussion forums, intranet, e-mail, and a VMT. The application of e-teaching tools for software application training focused first on application knowledge training but extended over time to include business context knowledge and collaborative task knowledge. In the beginning, the UML tool vendor and the virtual team responsible for the UML modeling tool's global deployment produced most of the learning content but over time the community using the technology also became a contributor.

The main contribution of this research was that several classes of e-teaching tools were used to support UML and UML modeling tool training but intranet and virtual meeting tool (VMT) were used to support UML and UML modeling tool training regarding all three types of knowledge (application knowledge, business context knowledge, and collaborative task knowledge). VMT was the most crucial class of tools because it not only contributed to the sharing of all three types of knowledge in the case company but also improved the users' motivation to use UML technology.

This study contributes to this thesis by providing new knowledge about UML modeling language and UML modeling tool teaching. Research results are relevant for IS researchers whose research interests are related to UML modeling, UML modeling tools, or e-teaching tools. Research topics may include, but are not limited to, research in the context of complex technology training, UML modeling language and UML modeling tool training, and e-teaching tools usage in information systems training. IT practitioners benefit from the results when they plan and deploy UML modeling language and UML modeling tools training.

2.5.4 Limitations and future research

The main limitation of this study is the study was conducted in only one organization. Future research is needed to confirm the results in the context of other organizations. Another limitation is only training organized through VMT was directly evaluated; specifically, skills, knowledge, and trainees' motivation were evaluated at the individual level. Other e-teaching tools' impact on trainees' skills, knowledge, and motivation were evaluated secondarily through measuring the popularity of each e-teaching tool. Therefore, research is needed to determine which e-teaching tools are the most effective at supporting learning of the UML modeling language and UML modeling tool in ways resulting in

learners satisfied with the training and the training positively impacts the skills, knowledge, and motivation of the learners.

The use of e-teaching tools evolved over time in the case company regarding the number of tools deployed and the coverage of not only the application knowledge training but also the business context knowledge and collaborative task knowledge training. The usage of various classes of e-teaching tools should be studied in future longitudinal studies to understand these phenomena better.

2.6 The case company and my role in the case company

All the studies presented were conducted in a global high-technology corporation in the business of developing products in multiple sites with multiple partners. Business units in the case company ran one or more product lines in which product programs produced product releases under the guidance of product roadmaps and release plans for customers in specific market segments. Product programs deployed software and hardware platform releases developed by internal platform units, inter-organizational consortiums, and external providers. Product programs were run either by the case company or its partners. The platform releases integrated hardware and/or software component releases developed internally or by partners or purchased off-the-shelf from external providers. Partners included OEMs, consortiums, outsourced software and hardware development, research centers, and open source communities. Requirements were collected from markets, service providers, and other internal and external sources.

IT support and development for applications used by R&D was organized by virtual teams consisting of personnel from the global IT department and the department responsible for process and information systems development and support for R&D as well as outsourced resources working for both the global IT department and the department responsible for process and information systems development.

For the duration of the studies, I was working in several roles within the case company (Table 2) in the department responsible for process and information systems development and support for R&D.

Because I had an official role in the case company, especially close attention was paid to verifying the results to ensure their reliability. In all the papers, there was at least one additional author who reviewed the same material and results, therefore ensuring conclusions were valid. As a part of this process, for example, content of the interviews was crosschecked and when needed more information was obtained if possible. More details of such actions taken are included in the papers.

TABLE 2 My roles in the case company.

2009-2012	**Senior Manager** responsible for architecture & system design process and operational development including process development, IT support and running process and development projects. Both direct line reports and outsourced resources.
2007-2008	**System Specialist** for UML modeling tool during global evaluation and deployment in the case company. Managing virtual team.
2006	**Maternity leave**
2004-2005	**System Specialist** for an information system which support integrated Requirements Management and Release Management System. Managing virtual team.
2003	**Maternity leave**
2000-2002	**Line Manager** for a team which was responsible for development, deployment and support of information systems used globally in R&D in the data warehouse, groupware, and portal technologies areas. Both direct line reports and outsourced resources.
1998-2000	**IT Project Manager** Planned, implemented and deployed a data warehouse solution for R&D globally; manage internal and subcontractor resources
1996-1998	**System Specialist** for information systems supporting R&D project management including deployment, development and support of ISs. Solutions created and supported in co-operation with process and concept owners, local support organization, IT specialists and subcontractors.

3 POST-EVALUATION OF STUDIES

In this chapter, I evaluate the relevance and rigor of each study in three respects. Firstly, I evaluate the relevance of each study from the perspective of the case company after the study was conducted, specifically, how the case company could utilize the results of the study. Second, I evaluate the relevance of each study based on how results were used in IS research. Third, I evaluate the rigor of the study from the perspective of methodological choices made during the study.

3.1 An Information Systems Design Product Theory for the Class of Integrated Requirements and Release Management Systems (Article I)

3.1.1 Relevance of research results for the case company

The research results of this study were the design product theory for the class of integrated requirements and release management systems (RRMS), including the requirements for RRMS instances and the design meeting the requirements. The design consists of an information model and the attributes for the elements presented in the information model.

This study provided relevant information for the company in which features of the information system provide critical support for the integrated requirement and release management process. After the study was completed, the case company decided to invest in a commercial system which would replace the existing in-house system. The commercial system evaluation, configuration and deployment was a large-scale migration project with a total investment of 10 million euros over five years. The requirements and design were reused during the evaluation, configuration, and deployment of the commercial system. The decision in support of replacement was made to enable usage of a commercial system instead of in-house development.

During the evaluation, configuration and deployment of the commercial system the information model was further developed. For the purposes of sharing and agreeing overall enterprise level architecture, the level of granularity in the information model was suitable. However, for the purposes of planning and implementing the information model through configuration of the commercial system, a more detailed information model was needed. Thus, the study's resulting information model was more suitable for the enterprise level architecture planning in the case company.

Interestingly, even if a new commercial system replaced the existing requirement and release management tool, UML modeling tool(s) continued to be used in the same ways. UML models were linked and imported into the system supporting the requirements and release management system. However, none of the business units or teams completed end-to-end models; specifically, they did not create complete models of the system. Thus, providing a new UML modeling tool and commercial system to replace the existing requirement and release management tool did not impact the way UML modeling was used in this regard.

3.1.2 Relevance of research results for science

Documented requirements and design represent the conceptual design IT artifact as described by Rossi and Sein (2003). According to March and Smith (1995), when building the *first* artifact, the research contribution lies in the novelty of the artifact and in the persuasiveness of the claims that it is effective. According to Käkölä et al. (2010), there are no requirements or design depicted for an information system supporting a requirement and release process. Therefore, the research can be considered novel. Additionally, the design was evaluated against the requirements and it fulfills the requirements. Therefore, the design artifact can be considered effective.

As the original paper was published several years ago, it is possible to evaluate how the study's results have been further developed. To find related studies, the following research portals were searched: Scopus, ACM Digital Library, IEEE Xplore Digital Library, Google Scholar, and ABI/INFORM (Proquest). Eleven studies containing references to the article were found. In two studies, the study's results were further developed. Tang and Liu (2010) proposed the definition and elements of a meta-requirement. Lu (2015) has extended the information model to cover test management. Thus, IS researchers have further developed the results provided.

3.1.3 Methodological Rigor

In the first paper, the design science research approach proposed by Walls et al. (1992) was applied. Our research question in this study was: "What are the meta-requirements and meta-design of Information System Design Theory for the class of integrated requirements and release management system (RRMS) in a globally distributed product development organization?" The main contribu-

tion in terms of Walls was the partial design product theory (ISDT) for the class of RRMS, including the meta-requirements for RRMS product instances and the meta-design that meets the meta-requirements. As there already existed partial ISDT provided relevant input for the study (i.e., ISDT for an information system supporting requirements management) continuation of the already existing ISDT development was relevant and therefore design research was conducted following Walls et al. (1992).

During the research process, some limitations during research implementation were found: 1) During the research process no evidence could be found regarding kernel theories availability in the context of the RRMS design. Thus, immature or missing kernel theories within the context of requirements and release management limited the creation of ISDT within this context. 2) The ISDT creation process was not very well phrased in the original paper and therefore required substantial effort to learn during the research process and explain later during the reporting phase. The second challenge is backed by Walls et al. (2004) who conducted a literature review and concluded the research method's usability and ease of use needs to be improved. In their study, Walls et al. (2004) concluded that there were only 26 papers applying their proposed research approach published during a 12-year period from the time that their original paper was published. Thus, this study represented a research approach which very few IS researchers have applied.

3.2 A Framework for Evaluating the Version Management Capabilities of a Class of UML Modeling Tools from the Viewpoint of Multi-site, Multi-partner Product Line Organizations (Article II)

3.2.1 Relevance of research results for the case company

The main contribution of this research was the set of validated evaluation criteria for the version management capabilities of UML modeling tools. The set of evaluation criteria and the evaluation results were used by the case company. These results provided relevant information for the case company regarding the differences that the UML modeling tools had in terms of version management. Additionally, the set of evaluation criteria was shared with the UML modeling tool vendor. The aim was the UML modeling tool vendor would develop the tool so that it supports all the evaluation criteria. At the time the original paper was written, the following evaluation criteria were not fulfilled by the UML modeling tool that was used in the case company:

- Availability of element level history (i.e., which modifications were done and who did them)
- Diagram and Element level branching (at the time of the writing branching was possible at the package and model levels only)

- Check in / Check out for diagrams and elements (at the time of writing Check in/ Check out was possible on model and package levels only).

Since 2015, the product has supported the availability of element level history. Thus, the tool has been developed to better fulfill the evaluation criteria and benefitted other people than just those at the case company.

3.2.2 Relevance of research results for science

According to Koivulahti-Ojala and Käkölä (2010), the literature does not provide a comprehensive set of evaluation criteria which could be applied in industrial settings to evaluate the version management capabilities of UML modeling tools in a globally distributed product development organization. Therefore, it can be claimed the set of evaluation criteria is novel. Furthermore, the set of evaluation criteria was successfully used for the evaluation of two UML modeling tools. According to the laboratory tests, it was possible to differentiate two UML modeling tools according to their version management capabilities.

Because the original paper was published several years ago, it is possible to evaluate how the study results have been further developed. To find related studies, the following research portals were searched: Scopus, ACM Digital Library, IEEE Xplore Digital Library, Google Scholar, and ABI/INFORM (Proquest). According to the search, eight studies were found with references to the article. However, none of these studies further developed or applied the evaluation criteria.

3.2.3 Methodological rigor

This study represented the evaluation phase of design science research (i.e. evaluation of existing UML modeling tools). According to March and Smith (1995), design science research consists of two basic activities: build and evaluate. Evaluation is the process of determining how well an artifact performs; it refers to the development of criteria and the assessment of artifact performance against those criteria.

The research approach was conceptual-analytical considering a new set of evaluation criteria was created. During the concept-analytical phase, relevant literature related to version management was analyzed, the requirements in the case company documented, and each criterion created and documented accordingly. The limitation is no literature review was conducted in the domain of version management during this phase. This decision was made based on my experience with several version management tools and knowledge that, at the time of writing, the version management capabilities of the tools were similar with each other. Thus, a literature review would not provide additional information.

However, a controlled test approach was applied when the set of evaluation criteria was assessed in a laboratory test. During laboratory testing, I installed all the needed software and tested the features. The rigor of the laboratory test was ensured by repeating the tests with two different versions of the tools.

In this study, only one domain of the UML modeling tool's features were studied (version management). These features were evaluated from the perspective of a globally distributed R&D organization where users of the UML modeling tool are product developers. The defined evaluation criteria scale was a binary "no" or "yes." Thus, it represented a qualitative evaluation for a limited user group and purpose. In general, evaluation of commercial off-the-shelf software is considered challenging. Wanyama and Far (2008) name multiple stakeholders and multiple objectives as challenges in the evaluation of commercial off-the-shelf software. Jadhav and Sonar (2009) state software evaluation is a multiple criterion decision-making problem (MCDM) and based on their literature review, the analytic hierarchy process has been widely used for evaluating software packages. The analytic hierarchy process was first introduced by Saaty (1999) and provides a comprehensive approach for software package evaluation. With multiple criteria or several stakeholders, I recommend considering such a process.

3.3 Design, implementation, and evaluation of a Virtual Meeting Tool-based innovation for UML technology training in global organizations (Article III)

3.3.1 Relevance of research results for the case company

The main contribution of this research was the design, implementation, and evaluation of a VMT-based training method for teaching UML modeling and UML modeling tool. Design and implementation of training was specified regarding content, organization of training, training materials, and trainers' skills and knowledge. Based on the evaluation, VMT was applied successfully in the case company for UML training regarding improved learner satisfaction, skills, knowledge, and motivation. At the time of the study, the case company was not committed to divulging information related to exact travel or training instructor costs due to contractual reasons where an outside travel agency and training provider were involved. However, the decrease in training costs can be measured by comparison. One set of on-line training organized for 20 participants cost 12% compared to face-to-face training with similar content. This only includes costs related to the training itself. Additionally, no traveling costs were assigned for on-line training for participants or instructors. Therefore, the total decrease in cost was even greater, although we do not know the detailed traveling costs due to contractual reasons. The cost decrease of 88% was significant for the case company as the training sessions were organized on a regular basis. During a six-month period (June 2010 - November 2010), 29 sessions were organized, and each lasted 1-2 hours. After three years, VMT was still extensively used for training. During a six-month period (June 2013 - November 2013), 29 sessions were organized, and each lasted 1-2 hours.

An ongoing need for training sessions was, according to the virtual team supporting the UML modeling tool, due to at least two reasons: 1) training sessions were organized in a way that they supported both novice and advanced learners and, therefore, even if novice users became more knowledgeable, they still find beneficial to join training sessions; and 2) there was an ongoing need for training because when users joined a new project, team or organization which used UML, they typically needed to learn additional information they were not already familiar with such as new types of diagrams.

As users gained more knowledge about the tool's capabilities, they did not request such features they now knew already exist. Before the training was introduced, users sometimes requested features the tool already had. Furthermore, they gained additional knowledge about the templates and other methods they can use to configure the tool's output and input. Thus, they did not need additional configurations to be implemented by a support team or the vendor. For example, during the training, users were trained how to publish their models in the intranet. As a result, there were fewer requests sent to the support team and tool vendor for new publishing capabilities. Users were also able to better formulate new requirements to improve the tool. For example, as a part of the training, they learned how to use version management. However, as they become more familiar with version management they can suggest new requirements for it.

3.3.2 Relevance of research results for science

According to a literature review conducted by (Koivulahti-Ojala and Käkölä, 2012), until 2012, there was only one paper published where the adoption of UML modeling training in industrial settings via e-teaching tool was studied, which was by Bunse et al. (2006). The limitation of that study is the training method was not described and the training did not cover UML modeling tool training. Therefore, I can claim that the training method is novel for the IS research community. Furthermore, the training method was successful regarding improved skills, knowledge, and motivation of users as well as learners' satisfaction.

Since the original paper was published several years ago, it is possible to evaluate how the study results have been further developed. To find related studies, the following research portals were searched: Scopus, ACM Digital Library, IEEE Xplore Digital Library, Google Scholar, and ABI/INFORM (Proquest). According to results from the search, one study citing this article was found. That study is included in this thesis (Article V). This study provided further information regarding how this training method was developed over time.

3.3.3 Methodological rigor

In this study, a systematic literature review was conducted. It followed Kitchenham et al.'s (2009) approach. The literature review was used to assess the cur-

rent state of knowledge regarding UML and UML modeling tool training. Boell and Cecez-Kecmanovic (2015) propose that the systematic literature review is suitable only for a meta study summarizing the evidence from earlier research. As our aim was to assess the current state of the research by summarizing the earlier research, I claim that use of Kitchenham et al. (2009) was relevant.

In this study, the six-phased design research methodology presented by Peffers et al. (2007) was deployed. The result of the study was a training method. Methods represent one type of artifact that is a possible result. By definition, design research is about artifacts (i.e. artefacts are the final results of design research process). March and Smith (1995) identify four types of IT artefacts: constructs, models, methods, and instantiations. They define these as follows: "As in natural science, there is a need for a basic language of concepts (i.e., constructs) with which to characterize phenomena. These can be combined in higher order constructions, often termed models, used to describe tasks, situations, or artifacts. Design scientists also develop methods, ways of performing goal-directed activities." Later, Hevner et al. (2004) adopt the same list of IT artifacts. Specifically, Hevner et al. (2004) state, "effective design-science research must provide clear contributions in the areas of design construction knowledge (i.e., foundations, system development methodologies, modeling formalisms, ontologies, problem and solution representations, design algorithms), and/or design evaluation knowledge (i.e., methodologies, new evaluation metrics)." March and Smith (1995) state constructs, models, and methods can be instantiated in specific products, or physical implementations. Rossi and Sein (2003) name potential products of design research conceptual designs (e.g., definition of relational model), methods (e.g., design patterns), models and systems (e.g., prototypes and commercial applications), and better theories (e.g., relational algebra). Hevner et al. (2004) pointed out that IT artifacts constructed in design science research are rarely full-grown information systems used in practice. System development methodologies, design tools, and prototype systems (e.g., GDSS, expert systems) are examples of such artifacts. For this study, the definition presented by March and Smith (1995) of methods as ways of performing goal-directed activities was adopted.

The value of an artifact lies in its utility (March & Smith, 1995; Hevner et al., 2004). The utility of the method was mainly evaluated from the perspective of the individual learner - are the learners satisfied with the training and are their knowledge, skills, and motivation improved. Moreover, this information is also relevant to trainers, R&D management, and IT management for decision making regarding training. However, the main target was evaluating the utility of the new training method from the perspective of the learner.

In this study, learning considered a transformative process where, through learning, the initial state in the learner's mind is transformed to the new state which is different from the initial state if learning has occurred (Järvinen, 1999, p.3; Aulin, 1982, p. 15). Thus, we assumed it is possible to evaluate learners' skills, knowledge, and motivation after the training and learning may improve them. However, our focus has been on transformation in knowledge, skills, and

motivation through learning only. Kang and Santhanam (2003) identified three knowledge domains IS training programs should cover: application knowledge covering commands and tools embedded in IS applications; business context knowledge covering the use of IS applications to effectively perform business tasks; and collaborative task knowledge covering task interdependencies between various actors and how the IS application coordinates and mediates these interdependencies. According to Kraiger et al. (1993), training can positively affect individuals' motivation and therefore improvement in motivation was considered relevant to the study. However, this view can be considered too narrow and there are studies aimed at a more comprehensive approach for understanding and evaluating e-learning (e.g., Koponen, 2008). Learning is closely related to the understanding of data, knowledge, and information (Hälinen, 2011, p. 6). Despite its limitations, this study provided new information for both researchers and practitioners.

3.4 A lightweight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool (Article IV)

3.4.1 Relevance of research results for the case company

The main contribution of this research was the design, implementation, and evaluation of a new measurement instrument to evaluate users' satisfaction with the tool and services supporting the tool. Continuous user satisfaction measurement was relevant for the case company since in the case company there was a target to periodically measure both user satisfaction and service quality of the tools used in the case company. This was a mandatory action because the case company was committed to fulfill criteria set in ISO 9000 to maintain customer satisfaction. User satisfaction surveys were conducted six times between 2009 and 2013 but the same instrument was not used each time. Instead, the guidelines and instructions given inside the company were followed. Questions were slightly different each time and the results were not fully comparable with previously conducted surveys. However, the questions represented system and service quality-related questions. The number of questions decreased during a five-year period, however, the same instrument was also used for systems other than the UML modeling tool. The potential number in the target group was hundreds of users during the period of 2009 - 2013.

One open question was included in the survey. The results of the surveys as well as answers to the open questions were source for feedback for the tool vendor as well as for the continuous development of the service. Action plans were created after each time the survey was conducted and contained new initiatives such as new requirements for the vendor or any tasks aiming at developing the service.

3.4.2 Relevance of research results for science

Analysis results from the two surveys, conducted in a case company, indicate the new measurement instrument has adequate reliability and validity. There are existing instruments to measure end user computing satisfaction (EUCS) and service quality. I compare the new instrument to the two most widely used existing instruments from the perspective of using those on a regular basis for a considerable number of applications and users. The most widely used instruments for user satisfaction measurement are EUCS and UIS and for service quality measurement, SERVQUAL (Petter et al., 2008). As the EUCS instrument (Doll and Torkzadeh, 1988) contains fewer items compared to the UIS, even its short form (Ives et al, 1983; Baroudi & Orlikowski, 1988). I used the EUCS instrument for comparison purposes as it contains fewer items. EUCS deploys 12 questions to measure user satisfaction. IS ZOT SERVQUAL (Kettinger and Lee, 2005) deploys 54 additional questions to be answered for IS service quality. In total, if using both these instruments, there are 66 questions users need to answer. Thus, one user for one application using the new instrument answers 8 questions rather than 66 questions using the existing instrument. In the case company, the number of R&D users was 6 000 during 2008 (Käkölä et al, 2010). If we assume a user uses the same amount of time to answer to each question, the time used for answering questions using the new instrument is 88% lower compared to time spent answering questions using the existing instrument. If each user uses 5 seconds to answer each question and the response rate is 20% (number of users that answer to the survey), the total time spent in the user population using the new instrument is 13 hours and with the existing instrument it is 110 hours. Moreover, one user typically uses several systems, thereby multiplying the number of surveys he or she would have to complete. Additionally, the case company collected this information twice a year. Thus, the estimated time savings is in the hundreds of hours. Thus, for the research community, this instrument provides new information about an instrument which is more feasible for long term use from the perspective of the industry and saves a considerable amount of time in regular use compared to the most widely used existing instruments. For the case company, the difference was considered so relevant they chose not to use these existing instruments at all even though they were introduced for the case company.

As the original paper was published several years ago, it is possible to evaluate how the study results have been further developed. To find related studies, the following research portals were searched: Scopus, ACM Digital Library, IEEE Xplore Digital Library, Google Scholar, and ABI/INFORM (Proquest). According to the search, six studies contained references to the article. Metrailler and Estier (2014) referred to this study as evidence it is in business management's best interests to understand users' satisfaction and service quality during tool deployment. Islam (2011) referred to this study as one of the recent studies regarding user satisfaction. Gahalaut and Käkölä (2010) referred to this study to show tools supporting their assertion that software product

lines should provide adequate speed and be easy to use. Thus, this study represented empirical evidence of why and how user satisfaction and service quality are measured. Other studies (Koivulahti-Ojala and Käkölä, 2012; 2014) focused on the results from the case company's perspective.

3.4.3 Methodological rigor

The main contribution of this research was the design, implementation, and evaluation of a new measurement instrument to evaluate users' satisfaction with the tool and services supporting tool. This new measurement instrument represented a multi-item measurement instrument with two reflective constructs and these were user's satisfaction with the tool and services supporting the tool. According to Petter et al. (2007), a reflective relationship exists between a construct and measurement items when items are a reflection of the construct. They also name examples such as perceived ease of use, perceived usefulness, and satisfaction. According to Petter et al. (2007), formative constructs occur when the items describe and define the construct. One of their examples is that of organizational performance and how it is operationalized through three measures: productivity, profitability, and market share.

In this study, the research process followed instructions presented by Churchill (1979) for new instrument creation. Later, MacKenzie et al. (2011) proposed a 10-step process for the development of valid scales which guides both the instrument development and construct validation.

3.5 Training people to master complex technologies through e-Learning: Case of UML technology training in a global organization (Article V)

3.5.1 Relevance of research results for the case company

In the case company, different e-teaching tools had been used routinely in meetings for several years. Later, those tools were deployed for UML and UML modeling tool training. Based on this case study, face-to-face training and support were accompanied by a wide variety of e-teaching tools including Wikis, discussion forums, intranet, e-mail, and a virtual meeting tool. The application of e-teaching tools for software application training focused first on application knowledge training but extended over time to include business context knowledge and collaborative task knowledge.

The main results for the case company were that 1) e-teaching tools are suitable for teaching complex technologies and 2) tools routinely used for meeting purposes are also suitable for e-teaching. The company has used the same tools for teaching not only UML modeling and UML modeling tool technology but also for other complex technologies such as use of product lifecycle man-

agement (PLM) or enterprise resource planning (ERP) systems. Specifically, in the case company, project management was not previously supporting the use of VMT for teaching complex technology as there were concerns that teaching application knowledge concerning command level skills is not possible using VMT (i.e. commands/keystrokes needed). However, based on the study, users have different strategies to learn command level skills such as writing their own notes. Therefore, there were fewer concerns concerning implementing VMT for teaching complex technology.

3.5.2 Relevance of research results for science

According to the literature review conducted by Koivulahti-Ojala and Käkölä (2014), there are no longitudinal studies on UML and UML modeling tool training. According to this longitudinal case study, intranet and virtual meeting tool were used to support UML and UML modeling tool training regarding application knowledge covering commands and tools embedded in the information system, business context knowledge covering the use of information systems to effectively perform business tasks, and collaborative task knowledge covering how others use the information system in their tasks.

This paper was recently published. No studies referring to this paper were found when Scopus, ACM Digital Library, IEEE Xplore Digital Library, Google Scholar, and ABI/INFORM (Proquest) were searched.

3.5.3 Methodological rigor

This study was a case study following guidelines given by Yin (2003). In this study, it was important to prepare a case study database as I hold a managerial position within the company. The second author reviewed the case study database and ensured there is no bias due to the involvement of the first author in the daily activities of the case company. Additionally, members of the support team also reviewed and commented on the analysis results.

In the original paper, the term e-learning tool was used instead of e-teaching tool which has been used in this summary. At the time of writing, the most commonly used term was selected to ensure the paper is easy to read. E-learning tool is a commonly used word and there are several books published which include the word e-learning tool in the title. However, I agree with the studies where learning is considered a phenomenon which can be facilitated by teaching - it is the human being who has the capability to learn and teaching or training is considered a way to facilitate learning (Järvinen, 1999, p.3; Aulin, 1982, p. 15). Therefore, in this summary, the term e-teaching tool is used instead of e-learning tool. The e-teaching tool term is utilized to describe tools capable of being used to deliver training electronically such as a virtual meeting tool (VMT), Wikis, and e-mail.

4 THE STUDY: UML MODELING TOOL IMPLEMENTATION IN A GLOBALLY DISTRIBUTED PRODUCT ORGANIZATION

In this Chapter, I describe how each study contributes to answering the main research question: How can a globally distributed product company where UML modeling activities are scattered across different locations and countries implement a UML modeling tool? A detailed description of each study can be found in Chapter 2 where each study is presented.

I start by presenting the stages of implementing the UML modeling tool in the case company, and this provides background information for the entire study. Next, I present the schedule of the studies and summarize the main results. Finally, I compare the stages of the UML modeling tool implementation in the case company to those introduced by Jadhav and Sonar (2011) and propose a new stage model in the context of UML modeling tool implementation.

4.1 UML modeling tool implementation in the case company

In this Subsection 4.1, I describe how the UML modeling tool was implemented in the case company and how each study was related to the UML modeling tool implementation. The description covers both the UML modeling tool implementation project and use phase.

The UML modeling tool implementation has been documented based on the internal material (steering group meeting memos, project team meeting memos, requirement documents, training materials, and e-mails); studies reported in the five articles as referred to by article number; and in Chapter 3, documented experiences in the case company after the study was completed. This chapter has been reviewed together with two project team members to ensure it depicts the UML modeling tool implementation accurately.

The case company was a global distributed high-technology corporation developing products in multiple locations. During the project, a commercial UML modeling tool was evaluated and implemented globally. The project was initiated due to management's desire for one globally available tool to enable modeling with standard notation because the UML modeling tool mainly used for UML modeling purposes did not support UML2 and SysML notations, and it was in the end of its lifecycle (e.g. the development of the tool was stopped and it had compatibility problems). Also, users were utilizing different tools for UML modeling (Article I). Potential UML tool users in the case company were considered architects and engineers. They were in different countries and working in several time zones and business units.

When the project was initiated, the case company had experienced failed software package implementation projects. Project failures included overrun of costs, delays in project deployment, or software not possible to implement at all due to technical problems. For these reasons, there were previous projects stopped without completion of the software package implementation.

The project was organized as follows (Figure 5). The project steering group was the architecture management team where the architecture decisions over different business units were made; they approved the scope, schedule, resourcing, and main deliverables. The project team consists of the global IT department, the department responsible for the process and tool development and support for R&D, and subcontractors working for these departments. Additionally, at least one architect was involved from each business unit for different tasks such as trial, pilot, and requirement management during the project. Architects joining the project contributed in different ways to the work. Some architects discussed the topic inside their business unit and actively sought further information within their business unit such as current tools used. Some architects considered themselves as experts and were actively looking for information outside the company without involving the potential users within their business unit. Thus, the potential users of the tool in different business units were treated differently. The project organized trainings and business unit representatives were asked to provide information regarding which users should be involved in the training. From some business units there were several representatives and from some there were very few or none.

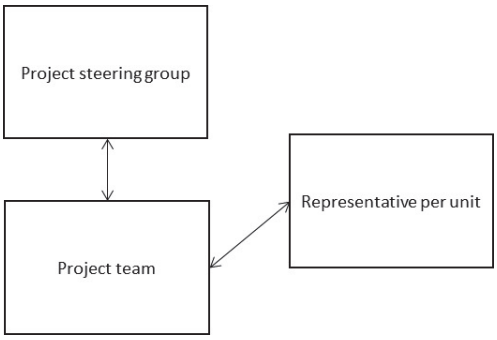

FIGURE 5 The project structure.

Details of UML modeling tool vendor liability, tool architecture, and financial evaluations are not provided in this thesis because the case company and tool vendors have agreed that the evaluation results are confidential.

Among members of the project team, there was discussion about past failures with software package implementation and therefore some guidelines were agreed upon. User participation was considered important and nominating a representative from each business unit was set as a target. It was considered a risk for the vendor to be involved the presentations to the steering group as they might try to sell the product. In this company, the new versions were called "promiseware" which indicated the new version is not real until it is available. In the past, a project had failed due to the vendor agreeing to provide new features but failed and the entire project was stopped due to a several months delay and the new release was not realized. Another important principle was the tools should be installed and used as early as possible so user feedback can be gathered before global implementation. In the past, there was a project where several months delay was realized when there were technical problems with installation. The project progressed following the case company's guideline for IT projects where the IT project is divided into five stages. The guidelines for IT projects followed a waterfall model, and therefore, the project manager together with project team tailored the IT guidelines. As a result, the project's stages were identified and steering group meetings held after the stages, but the content of the stages was tailored by the project team.

Each stage in the project lasted from weeks to several months (Table 3). Even though it was conducted in stages, the project did not follow a waterfall model where requirements are frozen in early stages of the project. Rather, requirements were managed in an iterative way. The project was initiated in October 2007 and it officially started in November 2007. Evaluation of the tool was completed in three stages. In Stage 1, the environment was created for trialing, the first set of requirements was created, and the list of potential UML modeling tools created. Based on this information, the decision was made by the steering group regarding which tools will continue to be installed during this stage. This was a conscious decision, and a risk was taken because after the next stage, there may be another tool under consideration. However, this approach enabled progress with hands-on activities. The selected tools for trial were installed (i.e. trial environment was created) and provided for users in the project team to use. Feedback based on usage of the tool was collected. During this stage, real data was entered (i.e. pilot environment was set up). In Stage 2, the pilot environment was provided for users on the project team to use. Again, feedback based on usage was collected. During Stage 1 and Stage 2, the list of requirements was revisited and update based on user feedback. Evaluation of the tools was completed and a decision about the UML modeling tool was made in May 2008 (Stage 3), and global implementation started for new teams during August 2008 (Stage 5).

In the case company, the project was considered successful because it was completed within the planned schedule and budget, and the UML modeling

tool can be used for the planned purposes by the user group. Three UML modeling tools used in the case company were replaced by this tool during the use phase. The case company was committed to continuous development of the tool and service.

TABLE 3 The stages in UML modeling tool implementation project.

Stage	Main Tasks	Timeline
Stage 1	Decision about those UML modeling tools which will be installed for trial purposes Environment setup for trial environment Initial list of requirements Initial list of potential UML Modeling Tools Demonstrating - the selected UML modeling tools demonstrated for users and feedback collected Pilot environment creation	Nov 2007 – Feb 2008
Stage 2	Piloting - the selected UML modeling tools demonstrated for users with real data and feedback collected Decision about the tool Service creation initiated	March 2007 - April 2008
Stage 3	Service creation continued	May 2008
Stage 4	Service creation finalized (documentation, administrative personnel training) End-user training Decision about the deployment	June 2008
Stage 5	Deployment including training and support	August 2008
Post-implementation	Continuous development of the tool together with vendor Continuous development of the service Continuous requirement management Evaluation of new features and new versions Integration of the tool to source code management and other systems Continuous training and support for the teams taking the tool into use, taking new features into use or extending their usage	September 2008 - 2013

The UML modeling tool use was supported after Stage 5 by a virtual team consisting of personnel from the global IT department, the department responsible for process and tool development and support for R&D, and subcontractors working for these departments. This team supported all the business units. No IT costs were assigned for individual users or business units (i.e. any license, server or other cost were managed centrally). The number of resources involved in support was three to five experts during the years 2009 - 2013 but not all of them were working full-time. The role of support personnel included: 1) a UML modeling and UML modeling tool expert from the tool vendor, technical support person(s) (1-2); 2) the person responsible for requirement management, user satisfaction survey, testing, business unit stakeholder management and training coordination; 3) a service manager; and 4) the team leader. The number of users was 1700 by the end of 2010 and 700 by the end of 2013 after major organizational changes. After 2013, due to company merger, usage of the selected UML modeling tool continued in two different companies and therefore are not reported as part of this study.

Modeling tool usage was voluntary as each business unit could decide on its own whether the UML modeling tool will be used. Additionally, in some business units it was agreed that teams and individuals themselves can make the decision. This lead in practice into a situation where there were individual users, teams, and business unit level evaluation on a continuous basis after the selection was made. Evaluation was realized in different teams and business units in different ways. The most comprehensive evaluation case included proof of concept creation during several meetings and evaluation of the tool capabilities to support modeling needs in the specific business unit. An initial meeting was held to review the modeling tool requirements. These requirements were related mainly to UML modeling (i.e. what is the best way to apply UML modeling). Experts from the vendor who know both the UML modeling tool and UML joined this meeting. During the meeting, the first version of the model was created and other requirements discussed. In the following meetings, the model was further developed, implementation of the UML modeling tool was planned, and any open issues in relation to the UML modeling tool or service related requirements were reviewed. Between the meetings, the model was further refined by a modeling expert, the support team prepared for training and deployment, and business users collected more input. The business unit representatives made a decision to begin using the UML modeling for modeling.

According to the study presented in Article I, UML models were created and used in the context of the requirement management and release management process. This approach to tool usage continued. During the use phase, the existing system to support requirement and release management was replaced, but the selected UML modeling tool continued to be used. Thus, approach using UML models that were linked and imported into the system supporting requirements and release management system was flexible in the sense it enabled changing the requirements and release management system without a

need to change the UML modeling tool. Additionally, for drafting purposes, other tools such as PowerPoint were still used. Diagrams were used to depict overall high-level architecture, subsystems, or components for relevant parts of the system. None of the business units or teams targeted for end-to-end modeling. Reverse engineering to create diagrams from code or headers of the code files was used in some of the business units and teams. Code generation or test automation was not used by any of the teams.

It required substantial effort from the support team to support those teams evaluating and adopting the tool into use. Typical adoption planning tasks included reviewing the needs for UML modeling, number of the users in the team, and previous experience with UML modeling and UML modeling tool. Typical adoption support included training, support for UML modeling (by vendor expert), user account creation, and modeling project setup. Because the company had very few people with extensive UML modeling knowledge at this phase, it was decided to pay for an expert from the tool vendor to support provide in case support was needed for UML modeling or UML modeling tool implementation.

In the case company, one full-time person was allocated to a role titled global concept owner during UML modeling tool evaluation, implementation, and support. This person was responsible for collecting user feedback and training, planning, and implementation during the project requirements management phase. After the deployment, this person was responsible for requirements management, conducting the user satisfaction surveys, and planning and implementing various training activities. I further elaborate how each of these tasks were conducted in the context of the UML modeling tool implementation in the following sections.

4.1.1 Requirements management

During the project, the requirements were collected and UML modeling tools evaluated against the documented requirements. Requirement management was facilitated by one person titled global concept owner. There were several sources of the requirements including relevant research and literature, standard requirements in the case company, requirements collected for other software packages in other software package implementation projects, and in-house systems within the case company as well as the project team. Interestingly, the source of the requirements were associated with a project team member or other individual rather than a specific document or web-page. Thus, the original references for literature or internet are not available. When the project was proceeding, the experiences gained during the trial and pilot phases were used as input for requirement management.

Some of the user representatives had used different UML modeling tools and had knowledge of the UML modeling language. Therefore, they primarily provided such requirements related to their experience when using UML or its modeling tool. Representatives from the support organization (from IT department and department responsible for the R&D process and tool development)

had experience from other evaluation projects and were aware of the standard requirements in the case company. Therefore, they mainly concentrated on relevant requirements based on their own experience or originated from other projects or standard requirements.

All the requirements were reviewed by the project team. In this way, all the project team members could contribute to the content. Requirements were managed in several iterations. After Stage 1, the number of requirements was 81. For each requirement, a priority was set by the project team. This prioritization included setting priorities to low, medium, or high from the users' perspective. This priority was considered specific for business and may conflict with the priority of other stakeholders such as the vendor or IT department. If business representatives had a different understanding of the priority during the project, the priority was negotiated and finally agreed upon one common priority. A target schedule was set for each requirement evaluation. Before making decisions about the tool, the project steering group reviewed the evaluation results.

In addition to the UML modeling tool-related requirements, there were 43 standard requirements. Standard requirements represented non-functional requirements related to technology, security, reporting, performance, and mobility. These were grounded on either known requirements in the globally distributed organization (e.g., network latency) or the case company's IT strategy (e.g., mobility). Interestingly, it was expected that a numeric result was possible for these standard requirements.

Requirements from other projects represented functional requirements from the perspective of globally distributed R&D. Examples of requirements from other projects were the meta-requirements and information model which had been reported for integrated requirements and release management system phases (Article I). Meta-requirements traceability, version management, and release management were considered relevant in this project. From the information model, the fields of history, origin, and workflow were seen appropriate to include.

Requirement management did not stop once the tool implementation project was completed. Already during the evaluation, all the requirements were provided for the vendor. Some were not fulfilled and therefore the follow-up of these continued. Additionally, new requirements were documented. Sources for the requirements included direct input from users, user satisfaction surveys, relevant literature, and IT's vision and strategy. Users' input for requirements included, but was not limited to, feedback send to support personnel by e-mail, user satisfaction surveys (Article IV), and sessions organized for local support persons where each local support person was asked to provide feedback regarding the tool and service. For the vendor, 70 requirements were reported during the years 2008-2012. These represented requirements for tool development or the service the vendor was providing. Furthermore, other input was given to the vendor including longer term IT strategies and visions when relevant as well as any plans related to UML and UML modeling tool usage when

appropriate. The main topic for further requirements was version management (Article II). During evaluation, project version management was considered as one requirement where the availability of version management capability in general was a requirement. No detailed requirements or evaluation criteria applicable to this context for version management were found in the literature. Once the tool was implemented, the support team became more experienced and individual teams in the case company started making use of version management. Therefore, more detailed evaluation criteria were developed (Article II) and shared with the vendor.

Additionally, requirements gathered during evaluation and deployment phases were re-used after the project ended. The UML modeling tool evaluated and deployed globally on a voluntary basis was later implemented for users that had previously used three other modeling tools, thereby replacing them. The requirements were re-used to provide evidence for the users of the existing tool regarding which requirements the tool can support. This reduced the resistance from users of the existing tools and saved time for the support team members as they could use the existing requirements.

4.1.2 Training

UML modeling tool training was organized as a classroom training using several e-teaching tools. During 2008, only class room trainings were organized. Later, a virtual meeting tool (VMT) based training was developed based on feedback from users in the form of user satisfaction surveys. Two user satisfaction surveys were conducted in 2009 (Article IV). The virtual support team analyzed the results of the surveys and concluded the instructions, user guides, and training practices had to be improved. It initiated several improvement activities during 2009. User satisfaction was improved after the VMT was deployed for the training and internet usage was enriched (Article III). In addition to the VMT, Wikis, intranet, discussion forums, and e-mail were used for the training.

4.1.2.1 Standard classroom training

Classroom trainings were organized by a tool vendor expert who had lengthy experience both in training and use of the UML modeling and UML modeling tool. The content of the training covered UML and UML modeling tool skills and knowledge. Classroom training sessions were either open for any potential user to join from any team or organized for a team planning to begin using the UML modeling tool. During 2009, a VMT-based training was developed and afterwards, classroom training sessions were organized only when a team was planning to deploy the UML modeling tool. However, only a few sessions were organized after 2009 when a VMT-based training was developed.

4.1.2.2 E-teaching

In addition to classroom training sessions, a variety of e-teaching tools including Wikis, discussion forums, intranet, e-mail, and a VMT were used in the case company (Article V, Table 4).

According to the study, the chosen tools were popular in the case company and improved user satisfaction with the UML tool. The case company used mostly e-teaching tools to support the application, collaborative task, and business context knowledge learning and sharing as called for by Kang and Santhanam (2003). According to the study, the VMT was the most crucial tool because it not only contributed to the sharing of all three types of knowledge but also improved the users' motivation to use to use the UML tool. VMT-based training sessions were organized using standard conference calls and a VMT. Most users had several years of experience in using both conference calls and VMT tools. During a six-month period (June 2010-November 2010), 29 sessions were organized, each lasting 1-2 hours. After three years, VMT was still used extensively for training. During a six-month period (June 2013 - November 2013), 29 sessions were organized, each lasting 1-2 hours.

Continuous need for trainings was, according to the virtual team supporting the UML modeling tool, caused by at least two reasons: 1) training sessions were organized in ways supporting both novice and advanced learners and therefore, even if novice users became more knowledgeable, they still find joining training sessions beneficial, 2) there was a continuous need for training because when users joined a new project, team, or organization which used UML, they typically needed to learn new information they were not already familiar with like new types of diagrams.

As users gained more knowledge about the capabilities of the tool they did not request features they know already exist. Before the training was introduced, users requested features the tool already had. In addition, they gained more knowledge about the templates and other methods they can use to configure the tool's output and input. Thus, they did not need additional configurations to be implemented by the support team or vendor. For example, during the training, users were trained how to publish their models on the intranet. As a result, fewer requests were sent to the support team and tool vendor for new publishing capabilities. Users were also able to better formulate new requirements to improve the tool. For example, as part of the training, they learned how to use version management. However, as they become more familiar with version management, they can suggest new requirements for version management.

Collaborative task and business context knowledge were mainly shared in sessions organized for local support persons who were responsible for supporting their teams in UML tool usage. As an example of business context knowledge sharing, local support persons shared their team's best practices using the UML modeling tool. Typically, 2-4 set of sessions were organized in a year. In addition, application knowledge was shared during these sessions covering new features, the tools' release schedules, new services, or planned changes in services such as the virtual team's contact information.

TABLE 4 E-teaching tools the virtual team applied for e-teaching in the case company (Article V).

E-teaching tool	Knowledge	Content
Wikis	Application knowledge	Sharing commercial plug-ins and related installation instructions/training materials
Wikis	Collaborative task knowledge	Sharing plug-ins made by users and related installation instructions/training materials
Intranet	Application knowledge	Self-study training materials Frequently Asked Questions New features of each UML modeling tool release Installation instructions How to apply to use the tool Recorded training sessions Material from other sessions
Intranet	Collaborative task knowledge	List of contact persons for teams using the UML tool Contact information for tool support team Recorded training sessions Material from other sessions
Intranet	Business context knowledge	Best practices in the form of business targets, way of using, UML modeling conventions, and deployment activities Recorded training sessions Material from other sessions
Discussion forum	Application knowledge	Share application knowledge with each other
Discussion forum	Collaborative task knowledge	Solving problems collaboratively
E-mail	Application knowledge	Informing all users about maintenance breaks, new features, training and other sessions to be organized
Virtual Meeting tools	Application knowledge	Training sessions Sessions where active users share best practices and application knowledge with other teams about applying the UML tool for modeling
Virtual Meeting tools	Collaborative task knowledge	Training sessions Sessions where active users share best practices with other teams about applying the UML tool for modeling including collaborative task knowledge
Virtual Meeting tools	Business context knowledge	Training sessions Sessions where active users share best practices with other teams about applying the UML tool for modeling including business context knowledge

4.1.3 User satisfaction measurement

In the case company, there was a target to periodically measure both user satisfaction and service quality regarding the tools used in the case company. This was a mandatory action as the case company was committed to fulfill the criteria set in ISO 9000 certification to maintain customer satisfaction. For the pur-

poses of measuring user satisfaction with the UML modeling tool and service, a new instrument was developed in 2009 (Article IV). There were three requirements for the instrument to be used as a measurement in the case company: 1) it should measure both the service quality and the user satisfaction regarding the tool; 2) there should be no more than 10 questions (including two standard questions of location and frequent of usage); and 3) the instrument should be applicable to further develop the service and the tool. The IS research community has delivered many comprehensive instruments to measure user satisfaction and service quality. However, the first requirement limited the choice to using an existing instrument as there was no instrument available to cover both the service quality and the tool-related satisfaction. Thus, a new instrument was created.

The new instrument was used two times in 2009. The virtual team supporting the UML Modeling tool analyzed the results. The team made decisions based on the means of all questions and the total mean of all questions. Based on the first survey, communication and training practices were improved because the means of questions related to instructions, user guides, and training were lower than the mean of all questions. Informative letters were emailed to the users, new guides were created, and the intranet pages providing information about the tool and related support were improved. The main improvement task was developing a new training method utilizing VMT (virtual meeting tool) (Article III). The results of the second survey revealed the improvements related to information sharing and training had raised user satisfaction and the availability and speed of the tool would be the next areas to improve (Article IV).

Later, user satisfaction surveys were conducted five times during the years 2009-2013 but not using the same instrument. Instead, the guidelines and instructions given inside the company were followed. Each time questions were different, and the results were not comparable with surveys conducted before. However, the questions represented system and service quality-related questions.

One open question was included in the survey. The results of the surveys as well as answers to the open questions were sources for feedback for the tool vendor as well as the continuous development of the service. Action plans were created after each survey was conducted. Action plans contained new initiatives such as new requirements for the vendor or any tasks aimed at developing the service. The main initiative during 2009-2013 based on user satisfaction survey results was the development of the VMT-based training (Article III).

4.2 Overview to the study

In this Subsection 4.2, I present the schedule of the studies and summarize the main results.

4.2.1 Schedule

The study process was comprised of five studies in which each research question comes from practice and provides new knowledge about UML modeling tool use, evaluation, or training (Figure 6). In Figure 6, each rectangle represents different studies. The left side of each rectangle marks the earliest point in time the data reported in the study covers and the right side of the rectangle marks the latest point in time the data reported in the study covers.

Article I provided insight regarding use of several UML modeling tools but the models were not complete. Specifically, it informed the case company that UML diagrams are created but employees are using different UML modeling tools. Thus, it provided a common need for the UML modeling tool implementation. During the five-year follow-up period reported in Chapter 3, no complete UML models covering all aspects of the system were created.

UML modeling tool evaluation, use, and training was studied in Articles II, III, IV, and V. In Article II the evaluation criteria for UML modeling tool version management in context of a globally distributed product development company were introduced. When a new UML modeling tool was implemented in the case company, it was relevant to measure users' satisfaction with the tool and the service (Article IV). This article provided new information for the research community regarding how users' satisfaction with the tool and service can be measured. Furthermore, when the results of the first survey were analyzed in the case company, it was concluded users are not satisfied with the training and the decision was made to implement a new training method utilizing a virtual meeting tool (VMT). This new training method was the subject of the study in Article III. Finally, Article V provided new information in the form of a longitudinal study regarding how the new training method utilizing VMT and other training methods evolved over time. Thus, Article V extended the study in Article III in two respects: several e-teaching tools were studied instead of one and the research period was extended to several years.

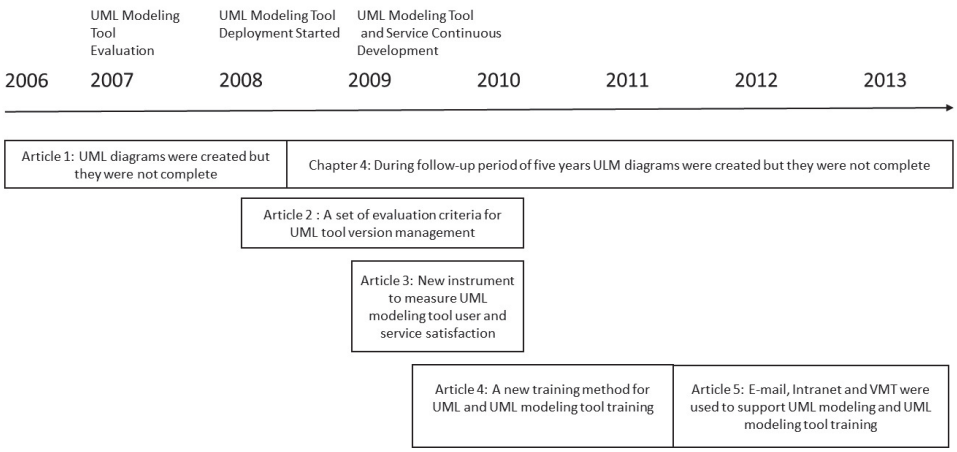

FIGURE 6 The schedule of the studies.

4.2.2 The main study results

In the following paragraphs, the main results are summarized based on the articles and summary of this thesis. The main results are presented in Table 5 with implications for practice and science as well as related evidence.

UML models were not complete according to the study which was conducted before UML modeling tool implementation evaluation, selection, and implementation started (Article I). UML modeling tool implementation did not change this. During implementation and post-implementation, no complete UML models were created and code generation was not in the scope of use. This result is complementary with Nugroho and Chaudron's (2008). According to their survey of 80 software professionals who use UML, they found most UML models did not cover all elements of the system.

In the case company, UML modeling tools' version management capabilities were evaluated and ranked to have the highest priority. Version management capabilities were implemented during the project and utilized by several teams in the use phase. Thus, version management capabilities can be considered vital in order to support modeling in a globally distributed product organization. After the tool was implemented, the support team became interested in regards to the most critical capabilities of version management. Therefore, more detailed evaluation criteria were developed (Article II). The extant literature does not provide a comprehensive set of evaluation criteria which can be applied in industrial settings to evaluate the version management capabilities of UML tools. The main contribution of this study was creating and evaluating a set of evaluation criteria.

When a new UML modeling tool was implemented in the case company, it was relevant to measure users' satisfaction with the tool and the service (Article IV). This article provided new information for the research community regarding how users' satisfaction with the UML modeling tool and service can be measured. Additionally, when the first survey's results were analyzed in the case company, it was concluded users were not satisfied with the training and it was decided to implement a new training method using virtual meeting tool (VMT). This new training method was the subject of the study in the Article III.

As UML is a complex language and users tend to begin using it gradually, continuous support and training was considered beneficial. Training and support were organized by a virtual team together with a UML modeling and UML modeling tool expert from a UML modeling tool vendor. Practitioners in globally distributed product companies may consider establishing a similar relationship to support UML modeling and UML modeling tool usage. Furthermore, according to the study reported in Article V, intranet and virtual meeting tools (VMT) were used to support UML modeling and UML modeling tool training in terms of application, business context, and collaborative task knowledge. Thus, intranet and virtual meeting tools (VMT) can be considered feasible for implementing e-teaching as they can be used to support teaching all three types of knowledge (application, business context, and collaborative task knowledge).

TABLE 5 The key results, related evidence and implications for science.

Related evidence	Implications for practice	Implication for science
Article I and follow-up during five years (Chapter 3)	UML models were created but they were not complete.	UML models were created but they were not complete. A complementary finding with Nugroho and Chaudron (2008).
Article II	Version management capabilities are required to support UML modeling in a globally distributed product organization. A set of evaluation criteria of the evaluation of the version management capabilities of the UML modeling tools.	A set of evaluation criteria for the evaluation of the version management capabilities of the UML modeling tools.
Article III and post-evaluation in Chapter 3	UML and UML modeling tool training can be organized through VMT cost efficiently so that users are motivated and their knowledge and skills are improved. According to the post-evaluation presented in this summary the training cost decreased in the case company by 88% per training session.	A training method to support UML and UML modeling tool training. The training method was described in terms of content, organization of training, training materials, and trainers' skills and knowledge.
Article IV	A lightweight measurement instrument, which can be applied to user and service satisfaction analysis for a UML modeling tool.	A lightweight measurement instrument, which can be applied to user and service satisfaction analysis.
Article V	Continuous support and training is beneficial as usage of the UML modeling tool evolves over time. Intranet and virtual meeting tool (VMT) can be used to support UML modelling and UML modeling tool training in terms of application, business context, and collaborative task knowledge.	Intranet and virtual meeting tool (VMT) were used to support UML modelling and UML modeling tool training in terms of application, business context, and collaborative task knowledge.

During the implementation of the UML modeling tool a new training method to support UML and UML modeling tool training was implemented. The training method was described regarding content, organization of training, training materials, and trainers' skills and knowledge in Article III. According to the post-evaluation presented in this summary in Chapter 3, the training cost

decreased in the case company by 88% per training session. Thus, it provided a significant decrease in the training costs. Moreover, due to a continuous need for training, practitioners in globally distributed product companies may consider establishing similar training.

4.3 Software package implementation stage model: Comparison and a new model

Software packages are vendor-developed software with the capability of being adopted by one or more customer organizations. Organizations have evaluated, selected, and used software packages since the 1990s when first software packages were introduced for the market. According to Gartner's IT Key Metrics Data (2012), 20% of companies' IT spending is on software. Thus, software package-related costs are a significant cost factor in companies' IT budgets. Evaluation and selection of a software package is considered a complicated and time-consuming decision-making process. There is a large body of research developing sophisticated methods and processes to help practitioners complete the evaluation, selection, and purchase processes. In this Subsection 4.3, I will compare the stages of the UML modeling tool implementation project presented in the Subsection 4.1 to the current body of software package implementation literature as summarized by Jadhav and Sonar (2011). Based on the results of a literature review, they present a methodology for selecting software. This methodology was chosen for comparison for two reasons. Jadhav and Sonar (2011) created the method based on existing literature and thus provides an overview rather than one more methodology; and, in the case company, it was expected the evaluation, selection, and purchase phases could be completed sequentially and it is possible to measure each attribute thereby reflecting similar thinking as Jadhav and Sonar (2011). Based on a literature review, Jadhav and Sonar (2011) identified six stages in the methodology: requirement definition, preliminary investigation of availability of software packages, short listing packages, establishing criteria for evaluation, evaluating software packages, and selecting the software package. They are presented in the following paragraphs and compared to the results of the stages presented in the Subsection 4.1.

1. Requirement definition

Identify functional and non-functional requirements of the software. According to Jadhav and Sonar (2011), the list of requirements must be accurate, complete and detailed.

2. Preliminary investigation of availability of software packages

Preliminary investigation of the availability of software packages that may be suitable candidates including investigation of major functionalities and features. Deliverable of this stage is a list of candidates.

3. Short listing packages

Candidate software packages identified in the second stage identified as not providing essential functionalities and features or does not work with existing hardware, operating systems, data management software, or network are eliminated.

4. Establishing criteria for evaluation

In this stage, criteria to be used for evaluating the software packages are identified and arranged in a hierarchical tree structure format. Each branch in the hierarchy ends in a well-defined and measurable basic attribute. The deliverable of this stage is a set of criteria arranged in hierarchical tree structure format.

5. Evaluating software packages

In this stage, metrics are defined and weights are assigned to each basic attribute in the criteria hierarchy. Rating is done against each basic criterion in the hierarchy for each software considered for detailed evaluation. An aggregate score is then calculated for each software package.

6. Selecting software package

The final stage is to rank the available alternatives in descending order by score and select the best software. They note the aggregate scores only give an idea about which software package is better over the other. Selecting the best software package is always a human-dependent process.

In the following paragraph, the methodology for selecting the software packages presented by Jadhav and Sonar (2011) is compared with the stages of the UML modeling tool evaluation and selection in the case company presented in Table 3. According to the study, requirement management was a continuous process. As more information was gained through usage of the different UML modeling tools or new information was gained, requirements were updated and new requirements may be added. Additionally, the candidates were eliminated as soon as there was enough information available regarding any sufficient reason to reject the tool rather than waiting for a comprehensive list of requirements to develop. Reasons for rejection were not limited to technical reasons. Thus, some tests were completed simultaneously rather than sequentially or continuously.

There were two documents corresponding to those described by Jadhav and Sonar (2011). These were the list of requirements and the list of available software packages. The list of potential software packages was created during the project and contained potential software packages available for UML modeling. The list of requirements was created in the first stage of the project but it continued evolving over time and not considered complete in any stage. Additionally, even though the aim was an exact numeric value representing the result of the evaluation, the project team concluded it is not possible to assign numerical values to some attributes. The priority was assigned in a manner which can be considered to resemble ranking using a scale of low, medium or

high. Thus, it was not possible to calculate aggregate values. Priority was considered specific for business and might conflict with the priority of other stakeholders such as the vendor or IT department. If business representatives had different understandings of the priority during the project, the priority was negotiated and finally agreed upon common priority. This was a limitation from the project perspective as it was not possible to show the different views of different stakeholders regarding the requirements. Using the analytic hierarchy process introduced by Saaty (1999) for analyzing better visibility for different stakeholder requirements could have been provided.

Based on the case study, the differences were considered positive, negative, or neutral compared to the stages proposed by Jadhav and Sonar (2011) (Table 6).

TABLE 6 Methodology for selection of the software packages by Jadhav and Sonar (2011) compared with results from this study.

Stage according to Jadhav and Sonar (2011)	Tasks in the stage according to Jadhav and Sonar (2011)	Results of this study (differences and considered impact for the selection of the tool)
Requirement definition	Identify functional and non-functional requirements of the software. According to Jadhav and Sonar, the list of requirements must be accurate, complete and detailed.	First list of requirements was generated in the first stage of the project but as new information and more experienced through usage was gained more details were added to existing requirements and new requirements added. Requirements were reviewed before the selection. Continuous development of the requirements was considered as positive as it enables more accurate and detailed requirements once more information was gained.
Preliminary investigation of availability of software packages	Preliminary investigation of availability of software packages that might be suitable candidates including investigation of major functionalities and features. Deliverable of this stage is a list of candidates.	Preliminary investigation of availability of software packages was completed at the same time as the first round of requirements gathering. The list of candidates was one of the results at this stage. Creating the short list as soon as possible was considered positive as it enables running activities in parallel.

(continues)

(TABLE 6 continues)

Short listing packages	Candidate software packages identified in the second stage that does not provide essential functionalities and features or does not work with existing hardware, operating system, data management software, or network are eliminated.	As the number of packages was high and substantial effort needed to gather information, in each stage those were eliminated, of which there was enough information for elimination to ensure progress in the project rather than waiting that all information is gathered for decision making. Elimination of the software packages which did not meet the requirements as soon as possible was considered positive as it enable running activities in parallel.
Establishing criteria for evaluation	In this stage criteria to be used for evaluation of the software packages are identified and arranged in hierarchical tree structure format. Each branch in the hierarchy ends into well-defined and measurable basic attribute. Deliverable of this stage is set of criteria arranged in hierarchical tree structure format.	Requirements were arranged into features and requirements. Each requirement had business priority which was negotiated result between the business representatives in the project team and the stage in the project that this requirement is planned to be reviewed. No measurable basic attribute was defined neither different stakeholders managed which was a limitation. Different stakeholder views were to some extent communicated to the steering group verbally. Considered as negative as not possible to provide different stakeholder views.
Evaluating software packages	In this stage metrics are defined and weights are assigned to each basic attribute in the criteria hierarchy. Rating is done against each basic criterion in hierarchy for each software considered for detailed evaluation. Aggregate score is then calculated for each software package.	Business priority of the requirement was considered as the weight (High, Medium, Low). Rating was done by written results based on usage of the tool during the project. No aggregate results were calculated. Considered as negative as not possible to provide different stakeholder views.
Selecting software package	The final stage is to rank the available alternatives in descending order of the score and select the best software. They note that aggregate scores only give an idea about which software package is better over the other. Decision of selecting best software package is always human dependable.	Requirements updated based on feedback were provided with business priority for the steering group as one input for final decision making. Neutral.

I further propose a new stage model (Figure 7) wherein the aim is making modifications to the stage model which will allow retaining changes considered positive during the case study. Due to these changes, some activities can be run in parallel.

4.3.1.1 Requirement definition

Identify the functional and non-functional requirements of the software. The list of requirements can be modified based on new information gained during installation and use of the software packages and on the preliminary investigation of availability of software packages. The list of requirements should be relatively mature before beginning to establish the criteria for and evaluation of software packages.

4.3.1.2 Preliminary investigation of availability of software packages

Preliminary investigation of availability of software packages that may be suitable candidates including investigation of major functionalities and features. Deliverable of this stage is a list of candidates. Can be conducted in parallel with requirement definition.

4.3.1.3 Selecting software package

As soon as there is enough information to eliminate a candidate from a short list, it can be done. This may include reasons such as the software package does not provide essential functionalities and features or does not work with existing hardware, operating systems, data management software, or networks. Input for decision making can come from the literature, vendor presentations, or installation and use of the software package. The final stage may include ranking the available alternatives in descending order based on the score and selecting the best software.

4.3.1.4 Establishing criteria for evaluation

In this stage criteria to be used for evaluation of the software packages are identified and arranged in hierarchical tree structure format. Each branch in the hierarchy ends in a well-defined and measurable basic attribute. The deliverable of this stage is a set of criteria arranged in hierarchical tree structure format.

4.3.1.5 Evaluating software packages

In this stage, metrics are defined and weights are assigned to each basic attribute in the criteria hierarchy. Rating is done against each basic criterion in the hierarchy for each software considered for detailed evaluation. An aggregate score is then calculated for each software package.

4.3.1.6 Installation and use of software packages

Installing software packages within the same environment it is going to be used in provides more information about the maintenance and use of the tool which can provide input for requirement definition, establishing criteria for evaluation, and evaluation software packages. If installation and use of the

software packages is not possible, experiences from existing users can be requested and collected, or the vendor can provide presentations and demonstrations.

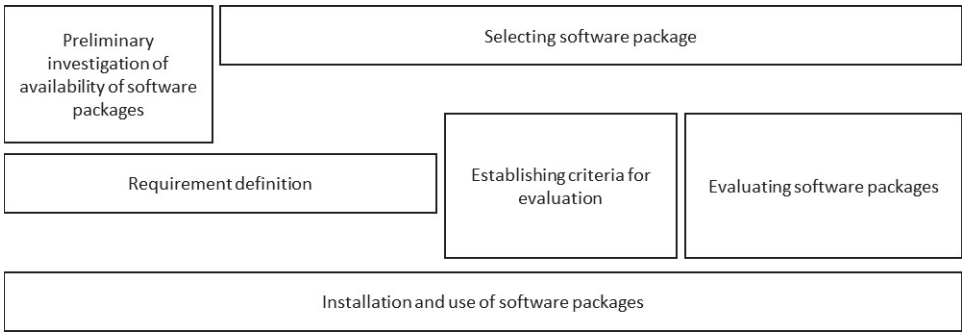

FIGURE 7 Proposed stage model for UML modeling tool selection.

5 DISCUSSION

This thesis offers new knowledge about UML modeling tool use, evaluation, and training. The main research question was: How can a globally distributed product company where UML modeling activities are scattered across different locations and countries implement a UML modeling tool? The research process was comprised of five studies wherein each research question comes from practice and provides new knowledge about UML modeling tool use, evaluation, or training in a globally distributed product company. In this chapter, I present these studies' most important implications for science and practice. Additionally, I present the limitations as well as some suggestions for future research.

5.1 Implications of results to science

In Subsection 5.1, I present each study, describe the scientifically novel findings, the findings that support the earlier results, and those that contradict earlier results. The summary of implications for science is listed in Table 7.

5.1.1 An Information System Design Theory (ISDT) for the class of requirements and release management systems (RRMS)

Globally distributed product development companies need to collect, analyze, and utilize requirements. Well-defined requirements are prerequisites for effectively scoping the product development projects and assigning them to internal units and partners. Integration of requirements and release management facilitates the end-to-end traceability of the distributed product development process from requirements to implementation. The research question was: "What are the necessary and sufficient properties for an IS which supports integrated requirement and release management processes in globally distributed product development?" The main contribution of this study is the design product theory for the class of RRMS, including the requirements for RRMS instances and the design that meets the requirements. The design consists of an information mod-

el and the attributes for the elements presented in the information model. According to Käkölä et al. (2010), there are no requirements or design depicted for an information system supporting the requirement and release process. Therefore, I claim the research was novel.

Moreover, our study provides additional information regarding how UML modeling was focused. According to our study, priorities, schedules, and other information stored concerning requirements, features, and releases were used to focus UML modeling efforts. Thus, this study provided new information about the UML and UML modeling tool usage in the context of the requirement and release management process.

TABLE 7 The key findings and their contribution for science.

Article	Implications for science	Contribution
Article I	A partial Information System Design Theory for the class of RRMS (requirements and release management systems), including the requirements for RRMS instances and the design that meets the requirements. The priorities, schedules, and other information stored in RRMS were used to prioritize the UML modeling efforts.	A novel finding
Article I	UML models were created but they were not complete.	A complementary finding with Nugroho and Chaudron (2008)
Article II	A set of evaluation criteria for the evaluation of the version management capabilities of the UML modeling tools.	A novel finding
Article III	A training method to support UML and UML modeling tool training. The training method was described in terms of content, organization of training, training materials, and trainers' skills and knowledge.	A novel finding
Article IV	A lightweight measurement instrument, which can be applied to user and service satisfaction analysis	A complementary finding
Article V	Intranet and virtual meeting tool (VMT) were used to support UML modelling and UML modeling tool training in terms of application, business context, and collaborative task knowledge.	A novel finding

According to the study, some UML models were created in the context of a requirement and release management process but they did not cover the whole system. This result complements Nugroho and Chaudron's (2008) study.

According to their survey of 80 software professionals using UML they found most of the UML models did not cover all elements of the system. Results of our study are thus in line with their study. According to our study, even though UML models were created, they were not complete.

5.1.2 A set of evaluation criteria for UML modeling tool version management

UML modeling tools' version management capabilities are critical when parallel and geographically distributed modeling activities need to be managed. The extant literature does not provide a comprehensive set of evaluation criteria capable of being applied in industrial settings to evaluate the version management capabilities of UML tools. The research question was: "What are the necessary and sufficient properties for version management to support UML modeling in globally distributed product development?" The main contribution of this study was creating and evaluating a set of evaluation criteria. According to Koivulahti-Ojala and Käkölä (2009), the literature does not provide a comprehensive set of evaluation criteria capable of being applied in industrial settings to evaluate the version management capabilities of UML modeling tools in the context of globally distributed product development. Therefore, I claim this set of evaluation criteria is novel.

5.1.3 A new training method to support training of UML and UML modeling tool

End-user training is complicated to implement in a globally distributed product development company where activities are scattered across multiple sites. Virtual meeting tools (VMT) enable synchronous communication globally through audio, chats, video, and sharing presentations. They provide a potentially cost-effective way to train large numbers of people in global settings. The research question was: "Can the UML and UML modeling tool training be organized and delivered through a VMT so that learners are satisfied with the training and the training positively impacts their skills, knowledge, and motivation?" The main contribution of this research was the design, implementation, and evaluation of a VMT-based training method for teaching UML and the features of a UML modeling tool. The training method was described regarding content, organization of training, training materials, and trainers' skills and knowledge.

According to the research, the VMT-based training positively impacted learners 'skills, knowledge, and motivation, and they were satisfied with the training. The training costs decreased in the case company by 88% per training session. Therefore, VMT-based training provided a cost-effective way to train users in using UML and the UML modeling tool. According to a literature review conducted by (Koivulahti-Ojala and Käkölä, 2012), there was, until 2012, only one paper published addressing adoption of UML modeling training in industrial settings via e-teaching tool, which was Bunse et al.'s (2006) study. The limitation of Bunse et al.'s (2006) study is that the training method was not

described and the training did not cover UML modeling tool training. Therefore, I claim the current study's training method is novel for the IS research community as it was described regarding training content, organization of training, training materials, and trainers' skills and knowledge, and it covered both UML and UML modeling tool training.

5.1.4 A lightweight measurement instrument, which can be applied to user and service satisfaction analysis for users of a UML modeling tool

The IS research community has delivered many comprehensive instruments to measure user satisfaction and service quality. However, they are tedious to deploy in industrial settings, possibly leading to low response rates. The research question was: "Is it possible to create a new adequately reliable and valid measurement instrument with eight items to measure both user satisfaction and service quality?" The main contribution of this research was the design, implementation, and evaluation of a new eight-item instrument to evaluate users' satisfaction with the tool and the services supporting its use. Analyzing the results of two surveys conducted in a globally distributed product development organization to measure user and service satisfaction of users using a UML modeling tool indicated the instrument has adequate reliability and validity.

This new survey instrument was compared to existing instruments. According to Petter et al. (2008), the most widely used instruments for measuring user satisfaction are EUCS and UIS and SERVQUAL for measuring service quality. For comparison purposes, I used the EUCS instrument as it contains fewer items compared to the UIS. If we assume a user takes the same amount of time to answer each question, the total time used for answering questions with the new instrument is 88% shorter compared to time spent answering the existing UIS and SERVQUAL instruments. Thus, for the IS research community, this study provides new information about a measurement instrument which is feasible from the industry's perspective in regular use for several applications as significantly less time is needed from users to answer the questions compared to existing instruments.

5.1.5 UML and UML modeling tool training through e-teaching tools: A longitudinal study

E-teaching tools facilitate asynchronous (e.g., Wikis) and synchronous (e.g., video-conferencing) learning. According to a literature review conducted by Koivulahti-Ojala and Käkölä (2014), there are no longitudinal studies on UML or UML modeling tool training via e-teaching tools in industrial settings. According to this longitudinal case study, an intranet and virtual meeting tool (VMT) were used to support UML and UML modeling tool training regarding application knowledge covering commands and tools embedded in the information system, business context knowledge covering the use of information systems to effectively perform business tasks, and collaborative task knowledge covering how others use the information system in their tasks. Furthermore,

Wikis, discussion forums, and e-mail were used to support UML and UML modeling tool training but not for all three types of knowledge. This research result is novel considering, according to our literature review, there are no longitudinal studies on UML modelling and UML modeling tool training.

5.2 Implications of results to practice

In this Subsection 5.2, I describe the studies' implications for practice. Each research question comes from practice and therefore many implications are comparable with the implications for science. However, the results are presented from the perspective of industry. For each study, I do not repeat the research questions as those were described in Subsection 5.1. The summary of the implications to practice is listed in Table 8.

5.2.1 An Information System Design Theory (ISDT) for the class of requirements and release management systems (RRMS)

The main contribution of this study is the design product theory for the class of RRMS, including the requirements for RRMS instances and the design meeting the requirements. The design consists of an information model and the attributes for the elements presented in the information model. Additionally, this study provided new information about the UML and UML modeling tool usage in the context of requirement and release management process.

Research results are relevant for R&D management who can take advantage of the results when planning, evaluating, implementing or deploying information systems to support the requirement or release management process together with the UML modeling tool. IT practitioners benefit from the results when planning, implementing, evaluating or deploying such systems.

5.2.2 A set of evaluation criteria for UML modeling tool version management

The main contribution of this research was the set of validated evaluation criteria for the version management capabilities of the UML modeling tools. Research results are relevant for R&D management and IT practitioners who can take advantage of the results when evaluating UML modeling tools' version management capabilities.

Global R&D organizations evaluating a UML modeling tool benefit from the framework as they can use it during the evaluation process or on the evaluation results of these two modeling tools. Especially for medium size companies, this is highly beneficial as it requires substantial effort to install the tools as well as complete the evaluation. The total effort required for both installation and evaluation was several man-months.

TABLE 8 The key implications for practice.

Article	Research results	Implication	Target group
Article I	A partial Information System Design Theory for the class of RRMS (requirements and release management systems), including the requirements for RRMS instances and the design that meets the requirements. Design includes information model and attributes for the elements presented in the information model.	Planning, implementing, evaluating or deploying an information system supporting requirement or release management process together with UML modeling tool.	R&D management, IT practitioners
Article II	A set of evaluation criteria for the evaluation of the version management capabilities of the UML modeling tools.	A set of evaluation criteria and results of evaluation can be used during the evaluation of UML modeling tool version management capabilities.	R&D management, IT practitioners
Article III	A training method to support UML and UML modeling tool training.	Planning, implementing, and evaluating UML modeling language and UML modeling tool training.	R&D management, IT practitioners
Article IV	A lightweight measurement instrument, which can be applied to user and service satisfaction analysis.	The measurement instrument can be applied to user and service satisfaction analysis.	IT practitioners
Article V	Intranet and virtual meeting tool (VMT) were used to support UML and UML modeling tool training in terms of application, business context, and collaborative task knowledge.	Intranet and virtual meeting tool (VMT) are suitable for teaching UML modeling and UML modeling tool.	R&D management, IT practitioners

5.2.3 A new training method to support training of UML and UML modeling tool

The main contribution of this research was the design, implementation, and evaluation of a VMT-based training method for teaching UML modeling language and UML modeling tool. Design and implementation of training was specified in terms of content, organization of training, training materials, and trainers' skills and knowledge. IT practitioners benefit from the new training method when planning, implementing, and evaluating UML modeling language and UML modeling tool training. R&D management can take advantage of the results when planning, implementing, and evaluating UML modeling

language and UML modeling tool training. IT practitioners and R&D management can take advantage of the results when making decisions about VMT usage in UML modeling language and UML modeling tool training.

5.2.4 A lightweight measurement instrument, which can be applied to user and service satisfaction analysis for users of a UML modeling tool

The main contribution of this research was the design, implementation, and evaluation of a new measurement instrument to evaluate users' satisfaction with the tool and services supporting tool. IT practitioners benefit from the proposed instrument when measuring user satisfaction and service quality for information systems. The measurement instrument is short and easy to use.

5.2.5 UML and UML modeling tool training through e-teaching tools: A longitudinal study

According to this longitudinal case study, an intranet and a virtual meeting tool (VMT) were used to support UML and UML modeling tool training regarding application knowledge covering commands and tools embedded in the information system, business context knowledge covering the use of information systems to effectively perform business tasks, and collaborative task knowledge covering how others use the information system in their tasks. The main result from the practitioners' perspective is that these tools are suitable for e-teaching of UML and UML modeling tool usage for three types of knowledge. IT practitioners benefit from the results when they plan and deploy training for UML and UML modeling tools.

5.3 Limitations and future research

The main limitation is the empirical evidence was collected only in one organization in all five studies. Empirical evidence was collected in a global high-technology corporation, developing products in multiple sites with multiple partners. Future research is needed to validate the results of those studies in other types of organizations.

Different strategies were applied to enable better generalization of the results (Table 9). In Article I, a literature review was conducted before the case study started to develop preliminary meta-requirements and meta-design. As the case company had already successfully used the application for several years with different products, inter-organizational setups, and partners, and due to the literature review conducted before starting the study, we felt confident the information model is suitable for other globally distributed product companies. In Article III, the target group of the informants represented different backgrounds (novice and experienced users), different roles (architect, programmer), different functions (IT and product development) and different con-

tinents (Asia and Europe). In Article IV, the measurement instrument was designed to be generally applicable for evaluating a variety of systems and services and was tested two times.

TABLE 9 The key implications for science and suggested domain for generalization of the results.

Article	Implications for science	Means used to enhance generalization of results	Suggested domain
Article I	A novel Information System Design Theory for the class of RRMS, including the requirements for RRMS instances and the design that meets the requirements. The priorities, schedules, and other information stored in RRMS were used to focus UML modeling efforts.	Literature review and a case study in a globally distributed product organization	Globally distributed product development companies
Article III	A novel training method to support UML and UML modeling tool training.	Case study	UML modeling language and UML modeling tool training for different roles and with different knowledge and skills (novice and experienced users)
Article IV	A lightweight measurement instrument, which can be applied to user and service satisfaction analysis.	Two surveys conducted in the case company for users of a UML modeling tool	All information systems and related services

Another limitation is no cost data could be collected for use in any of the studies due to demands by the case company not to divulge any internal or third-party related costs such as travel. Availability of cost data would offer better possibilities to more deeply analyze the data and could enrich the research results in all studies. This limitation was applicable to all five studies.

For Article II, the limitation is due to basing the new set of evaluation criteria on analysis of existing research in SW version management and documented requirements for assets requiring management in product line organization based on the experiences in the case company. A suggestion for further research is developing a set of evaluation criteria theoretically based on product development company mission and control parameters.

For Article IV, the research process followed instructions presented by Churchill (1979) to create a new instrument. A suggestion for further research is to apply more recently published processes such as those described by MacKenzie et al. (2011) for the construct validation.

For Article V, the limitation is conducting the longitudinal study in only one organization. The usage of various classes of e-teaching tools should be studied in longitudinal studies within different organizations to better understand which e-teaching tools are used and why for UML modeling and UML modeling tool training.

YHTEENVETO (FINNISH SUMMARY)

Unified Modeling Language™ (UML) on kansainvälinen standardi mallinnuskielelle. UML-mallinnuskieltä voidaan käyttää ohjelmiston vaatimusten, arkkitehtuurin ja rakenteen kuvaukseen, ohjelmakoodin generointiin sekä testauksen automatisointiin. UML-mallinnusohjelman avulla voidaan tukea UML-mallintamista. UML-mallinnusohjelmalla käyttäjät voivat luoda ja ylläpitää UML-malleja, generoida koodia sekä luoda UML-malleja koodin perusteella. UML-mallinnusohjelmia käytetään tuhansissa tuotekehitys- ja ohjelmistoalan yrityksissä ympäri maailman.

Tämä väitöskirja sisältää viisi tutkimusta UML-mallinnusohjelman käytöstä, evaluoinnista ja koulutuksesta. Kaikki tutkimukset toteutettiin tuotekehitysyrityksessä, jossa UML-mallinnusohjelmaa käytettiin useilla paikkakunnilla eri maissa. Tutkimuksen päätutkimuskysymys on: ”Miten useilla paikkakunnilla toimiva tuotekehitysyritys voi ottaa käyttöön UML-mallinnuskielen ja UML-mallinnusohjelman?”.

Ensimmäisen tutkimuksen tulosten mukaan UML-malleja laadittiin osana vaatimustenhallinta- ja paketointiprosessia, mutta UML-mallit eivät kattaneet koko suunniteltavaa järjestelmää. Toisessa tutkimuksessa arvioitiin, mitkä ovat välttämättömät mutta riittävät ominaisuudet UML-mallinnusohjelman versionhallinnalle maantieteellisesti hajautuneessa tuotekehitysorganisaatiossa. Tämän tutkimuksen tärkein tulos on kriteeristö UML-mallinnusohjelman versionhallintaominaisuuksien arviointiin.

Kolmannessa tutkimuksessa selvitettiin, voidaanko UML-mallinnusohjelman ja UML-mallinnuskielen koulutus toteuttaa sähköisen kokousjärjestelmän avulla niin että koulutus vaikuttaa positiivisesti oppijoiden tietoihin, taitoihin ja motivaatioon ja oppijat ovat tyytyväisiä koulutukseen. Tutkimustulosten mukaan oppijat olivat tyytyväisiä koulutukseen ja oppijoiden tiedot, taidot ja motivaatio käyttää UML-mallinnusta ja UML-mallinnusohjelmaa paranivat koulutuksen avulla. Sähköisen kokousjärjestelmän avulla toteutetun koulutuksen kustannukset olivat 88% pienemmät kuin perinteisen luokkakoulutuksen.

Neljännessä tutkimuksessa kehitettiin mittari UML-mallinnusohjelman käyttäjän tyytyväisyyden mittaamiseen. Viides tutkimus oli pitkittäistapaustutkimus, jossa tutkittiin, mitkä e-oppimista tukevat järjestelmät sopivat parhaiten käyttöön UML-mallinnuskielen ja UML-mallinnusohjelman opetuksessa niin että suuri määrä oppijoita oppivat käytössä vaadittavat tiedot ja taidot. Tutkimuksen mukaan intranet ja sähköinen kokousjärjestelmä tukivat ohjelman käyttötaitojen, siihen liittyvien prosessien sekä ohjelman välityksellä tapahtuvan yhteistyön oppimisessa. Yritykset voivat hyödyntää tutkimusten tuloksia suunnitellessaan ja toteuttaessaan UML-mallinnuskielen ja UML-mallinnusohjelman käyttöönottoa.

REFERENCES

Adelson, B. & Soloway, E. 1985. The role of domain experience in software design. IEEE Transactions on Software Engineering 11 (11), 1351–1360.

Aulin, A. 1982. The cybernetic laws of social progress. Oxford: Pergamon Press.

Baroudi, J. J. & Orlikowski, W. J. 1988. A short-form measure of user information satisfaction: a psychometric evaluation and notes on use. Journal of Management Information Systems 4 (4), 44–59.

Benbasat, I., Goldstein, D. K. & Mead, M. 1987. The case research strategy in studies of information systems. MIS Quarterly 11 (3), 369–386.

Boell, S. K. & Cecez-Kecmanovic, D. 2015. On being 'systematic'in literature reviews in IS. Journal of Information Technology 30 (2), 161–173.

Booch, G. 1994. Object-oriented analysis and design with applications (2nd edition). Redwood city, CA, USA: Benjamin/Cummings.

Bryman, A. & Cramer, D. 1999. Quantitative data analysis with SPSS release 8.0 for Windows: For Social Scientists. New York: Routledge.

Budgen, D., Burn, A. J., Brereton, O. P., Kitchenham, B. A. & Pretorius, R. 2011. Empirical evidence about the UML: a systematic literature review. Software: Practice and Experience, 41 (4), 363–392.

Bunse, C., Grutzner, I., Peper, C., Steinbach-Nordmann, S. & Vollmers, C. 2006. Coaching professional software developers - an experience report. In Proceedings of the 19th Conference of Software Engineering Education and Training, IEEE.

Carmel, E. & Agarwal, R. 2002. The maturation of offshore sourcing of information technology work. MIS Quarterly Executive 1 (2), 65–78.

Churchill, G. A. 1979. A Paradigm for Developing Better Measures of Marketing Constructs. Journal of Marketing Research 16 (1), 64–73.

Delone, W. H. 2003. The DeLone and McLean model of information systems success: a ten-year update. Journal of management information systems 19 (4), 9–30.

DeLone, W. & McLean, E. 1992. Information systems success: the quest for the dependent variable. Information systems research 3(1), 60-95.

Doll, W. & Torkzadeh, G. 1988. The measurement of end-user computing satisfaction. MIS Quarterly 12 (2), 259–274.

Doll, W. & Torkzadeh, G. 1991. The measurement of end-user computing satisfaction: theoretical and methodological issues. MIS Quarterly 15 (1), 5–10.

Dori, D. 2002. Why significant UML change is unlikely. Communications of the ACM 45 (11), 82–85.

Fitsilis, P., Gerogiannis, V. C. & Anthopoulos, L. 2014. Role of Unified Modelling Language in software development in Greece - results from an exploratory study. IET Software 8 (4), 143–153.

Gahalaut, G. & Käkölä, T. 2010. Evaluation of Frame-and Feature-based Software Product Line Tools from the Viewpoint of Mass Customization by End Users. In Proceedings of the AMCIS 2010.

Hevner, A., March, S., Park, J. & Ram, S. 2004. Design science in information systems research. MIS Quarterly 28 (1), 75–105.

Hälinen, R. 2011. An Evaluation Method for Virtual Learning Applications. Ph.D. Thesis, University of Tampere, Finland.

Islam, A. K. M. N. 2011. Information systems Post-adoption satisfaction and dissatisfaction: A study in the e-learning context. In Proceedings of 15th Pacific Asia Conference on Information Systems (PACIS).

Islam, A. K. M. N., Koivulahti-Ojala, M. & Käkölä, T. 2010. A light-weight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool. In Proceedings of the AMCIS 2010.

ISO. 2010. ISO/IEC TR 19759:2005 - Software Engineering - Guide to the Software Engineering Body of Knowledge (SWEBOK), Geneva, Switzerland: International Organization for Standardization.

ISO. 2012. ISO/IEC 19505-2:2012 - Information technology - Object Management Group Unified Modeling Language (OMG UML) - Part 2: Superstructure, Geneva, Switzerland: International Organization for Standardization.

Ives, B., Olson, M. H. & Baroudi, J. J. 1983. The measurement of user information satisfaction. Communications of the ACM 26(10), 785-793.

Jadhav, A. S. & Sonar, R. M. 2011. Framework for evaluation and selection of the software packages: A hybrid knowledge based system approach. Journal of Systems and Software 84 (8), 1394-1407.

Jiang, J. J., Klein, G. & Carr, C. L. 2002. Measuring information system service quality: SERVQUAL from the other side. MIS Quarterly 26 (2), 145–166.

Järvinen, P. 1999. Oman työn analyysi ja kehittäminen. Tampere: Opinpaja Oy.

Järvinen P. 2012. On research methods. Tampere: Opinpaja Oy.

Kang, D. & Santhanam, R. 2003. A longitudinal field study of training practices in a collaborative application environment. Journal of Management Information Systems 20 (3), 257–281.

Kettinger, W. & Lee, C. 2005. Zones of tolerance: Alternative scales for measuring information systems service quality. MIS Quarterly 29 (4), 607–623.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J. & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. Information and software technology, 51(1), 7–15.

Kobryn, C. 2002. Will UML 2.0 be agile or awkward? Communications of the ACM 45 (1), 107–110.

Koivulahti-Ojala, M. & Käkölä, T. 2009. Framework for Evaluating the Version Management Capabilities of a Class of UML Modeling Tools from the Viewpoint of Multi-site, Multi-partner Product Line Organizations. In Proceedings of the 43rd Hawaii International Conference on Systems Sciences, IEEE Computer Society, Hawaii, USA.

Koivulahti-Ojala, M. & Käkölä, T. 2012. Design, implementation, and evaluation of a Virtual Meeting Tool-based innovation for UML technology training

in global organizations. In Proceedings of the 45rd Hawaii International Conference on Systems Sciences, IEEE Computer Society, Hawaii, USA.

Koivulahti-Ojala, M. & Käkölä, T. 2014. Training people to master complex technologies through e-Learning: Case of UML technology training in a global organization. In Proceedings of the AMCIS 2014.

Koponen, E. 2008. The development, implementation and use of e-learning: critical realism and design science perspectives. Ph.D. Thesis, University of Tampere, Finland.

Kraiger, K., Ford, J. K. & Salas, E. 1993. Application of cognitive, skill-based, and affective theories of learning outcomes to new methods of training evaluation. Journal of Applied Psychology 78 (2), 311–328.

Käkölä, T., Koivulahti-Ojala, M. & Liimatainen, J. 2011. An Information Systems Design Product Theory for the Class of Integrated Requirements and Release Management Systems. Journal of Software Maintenance and Evolution: Research and Practice 23 (6), 443–463.

Lange, C. F., Chaudron, M. R. & Muskens, J. 2006. In practice: UML software architecture and design description. IEEE Software 23 (2), 40–46.

Lee, A. S. 1989. A scientific methodology for MIS case studies. MIS Quarterly 13 (1), 33–50.

Lu, Y. 2015. An information system design product theory for the class of eSourcing requirements, delivery and completion management systems for eSourcing service providers. Ph.D. Thesis, University of Jyväskylä, Finland.

MacKenzie, S. B., Podsakoff, Ph. M. & Podsakoff, N. P. 2011. Construct Measurement and Validation Procedures in MIS and Behavioral Research: Integrating New and Existing Techniques. MIS Quarterly 35 (2), 293–334.

March, S. & Smith, G. 1995. Design and natural science research on information technology. Decision Support Systems 15 (4), 251–266.

Metrailler, A. & Estier, T. 2014. EVOLIS Framework: A Method to Study Information Systems Evolution Records. In Proceedings of the 47th Hawaii International Conference on Systems Sciences, IEEE Computer Society, Hawaii, USA.

Nugroho, A. & Chaudron, M. R. 2008. A survey into the rigor of UML use and its perceived impact on quality and productivity. In Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM.

Peffers, K., Tuunanen, T., Rothenberger, M. & Chatterjee, S. 2007. A design science research methodology for information systems research. Journal of Management Information Systems 24 (3), 45–77.

Petre, M. 2013. UML in practice. In Proceedings of the 35th International Conference on Software Engineering. IEEE.

Petter, S., DeLone, W. & McLean, E. 2008. Measuring information systems success: Models, dimensions, measures and relationships. European Journal of Information Systems 17 (3), 236–263.

Petter, S., Straub, D. & Rai, A. 2007. Specifying formative constructs in information systems research. MIS Quarterly 31 (4), 623–656.

Pitt, L. F., Watson, R. T. & Kavan, C. B. 1995. Service quality: A measure of information systems effectiveness. MIS Quarterly 19(2), 173–187.

Nunamaker Jr, J. F., Chen, M. & Purdin, T. D. 1990. Systems development in Information Systems Research. Journal of Management Information Systems 7(3), 89–106.

Rossi, M. & Sein, M. 2003. Design Research Workshop: A Proactive Research Approach. Presentation delivered at IRIS 26, August 9 – 12, 2003. http://tiesrv.hkkk.fi/iris26/presentation/workshop_designRes.pdf last accessed January 16, 2004.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorenson, W. 1991. Object-Oriented Modeling and Design. Englewoord Cliffs, NJ: Prentice-Hall.

Saaty, T. L. 1999. Fundamentals of the Analytic Network Process. In Proceedings of the 5th International Symposium on the Analytic Hierarchy Process (ISAHP).

Salo, A. & Käkölä T. 2005. Groupware support for requirements management in new product development. Journal of Organizational Computing and Electronic Commerce 15 (4), 253 – 284.

Smithson, S. & Hirschheim, R. 1998. Analyzing information systems evaluation: another look at the old problem. European Journal of Information Systems 7(3), 158–174.

Straub, D., Boudreau, M. C. & Gefen, D. 2004. Validation guidelines for IS positivist research. The Communications of the Association for Information Systems 13 (1), Article II4.

Symons, V. J. 1991. A review of information systems evaluation: content, context and process. European Journal of Information Systems 1 (3), 205–212.

Tang, Z., & Liu, S. 2010. The Constructing Method of Meta-requirement Analysis Model. In Proceedings of the 3rd International Conference on Business Intelligence and Financial Engineering (BIFE). IEEE.

Ulrich, K.T. 1995. The role of product architecture in the manufacturing firm. Research Policy 24 (3), 419–440.

Walls, J., Widmeyer, G. & El Sawy, O. 1992. Building an information system design theory for vigilant EIS. Information Systems Research 3 (1), 36–59.

Walls, J., Widmeyer, G. & El Sawy, O. 2004. Assessing information system design theory in perspective: how useful was our 1992 initial rendition. Journal of Information Technology Theory and Application 6 (2), 43–58.

Wanyama, T. & Far, B. 2008. An empirical study to compare three methods for selecting COTS software components. International Journal of Computing and ICT Research 2 (1), 34–46.

Yin, R. K. 2003. Case study research: Design and methods (3rd edition). Thousand Oaks: Sage Publications.

Yin, R. K. 1989. Case study research: Design and methods. Thousand Oaks: Sage Publications.

ORIGINAL PAPERS

I

AN INFORMATION SYSTEMS DESIGN PRODUCT THEORY FOR THE CLASS OF INTEGRATED REQUIREMENTS AND RELEASE MANAGEMENT SYSTEMS

by

Timo Käkölä, Mervi Koivulahti-Ojala & Jani Liimatainen, 2011

Journal of Software Maintenance and Evolution: Research and Practice vol 23, 443-463

JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. (2010)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.492
SOFTWARE PROCESS IMPROVEMENT AND PRACTICE ARTICLE

Research

An information systems design product theory for the class of integrated requirements and release management systems

Timo Käkölä[1,*,†], Mervi Koivulahti-Ojala[1] and Jani Liimatainen[2]

[1]*University of Jyväskylä, 40014 Jyväskylä, Finland*
[2]*Accenture, 00101 Helsinki, Finland*

SUMMARY

High-tech companies conducting product development need to collect and analyze requirements effectively, plan and implement releases, and allocate requirements to appropriate releases. Requirements and release management are complicated because development activities typically are scattered across multiple sites, involve multiple partners in different countries, leverage various development methods and tools, and are realized through various organizational arrangements such as release projects in organizations structured around products and permanent release teams in organizations responsible for the long-term development and maintenance of strategic software and hardware assets. Flexible, scalable, and secure groupware-based support for the activities provides substantial payoffs. Yet, the extant literature provides little theoretical guidance for designing and using requirements and release management systems (RRMS) in multi-site, multi-partner environments. This article develops the meta-requirements and a meta-design of an Information Systems Design Product Theory for the class of RRMS based on a case study in a global company and a literature review. The theory is scalable to meet the needs of global companies but simple enough so that small and medium-sized companies can also leverage it to implement requirements and release management solutions. Copyright © 2010 John Wiley & Sons, Ltd.

Received 27 November 2008; Revised 30 October 2009; Accepted 6 November 2009

KEY WORDS: global software product development; information systems design theory; knowledge management; release management; requirements management; software process improvement

*Correspondence to: Timo Käkölä, University of Jyväskylä, 40014 Jyväskylä, Finland.
†E-mail: timokk@jyu.fi

1. INTRODUCTION

To succeed in the global markets of software-intensive products, high-tech companies need to shorten the cycle time of new product development while improving the product quality and service delivery and maintaining or reducing the total resources required [1,2]. This concern can be dealt through (1) internal strategies such as global software development, where development resources are distributed globally to reap cost benefits, leverage specialized competencies, and address the specific needs of geographically defined markets [3–5], and software product line engineering and management, that is, the strategic acquisition, creation, and reuse of software assets [6–8] or (2) external strategies such as acquiring commercial off-the-shelf components and outsourcing software development, maintenance, and related services to best-in-class service providers [9,10].

All the strategies require companies to effectively collect, analyze, and utilize requirements [11–16]. This is particularly true during the earliest phases of product development in which different stakeholders need to integrate their knowledge into product concepts that direct the internal personnel and the service providers during the downstream phases of product development [17–19]. A well-defined product concept is necessary to establish a viable product line architecture that can be shared across the products within the product line to enable strategic reuse. Well-defined requirements, architectural interfaces, and product architectures are prerequisites for assigning appropriately scoped projects to internal units and service providers for implementation [9,10].

The achievement of such integration is complicated by several factors [20]. Numerous requirements ranging from abstract wishes to detailed technical solution proposals are created continuously. Development activities are scattered across many sites and partners in different countries, limiting the possibilities for setting up face-to-face meetings [14]. Organizational changes, differences in organizational cultures, and divergent perceptions about the prospective product's mission may make it difficult to reach an agreement about the product definition [21].

A commonly enacted software product line governance model and a strategic product line roadmapping process should be instituted to ensure that the organization is ready for multi-site development [8,22]. All sites should use compatible processes, methodologies, tools, and terminology as much as possible to enact the governance model [4]. Each product roadmap outlines the respective product line at a given point in time, explicates (from the market viewpoint) the major common and variable features of all foreseeable products of the product line, and schedules the deliveries of the products to markets [7, Chapters 2, 9]. For every product, a release plan should be made that allocates the features to scheduled product releases and responsible development organizations and documents the allocations.

Release planning must be conducted carefully and systematically by the stakeholders responsible for the requirements and the product strategy and by the internal and external stakeholders responsible for implementing the requirements in releases and the resulting release plans must be communicated clearly and in time to the stakeholders [23]. Otherwise, it is difficult for the providers to commit resources for scheduling and synchronization of their production activities to meet the requirements specified in the release plans. For example, the scopes of software releases cannot be measured in terms of the functional size [24,25] if the requirements are not linked to the releases implementing them because functional size measurement is solely based on the requirements.

A critical component of the governance model is that all requirements are (1) captured in a repository to ensure that they are neither missed nor overlooked and (2) subjected to effective

DOI: 10.1002/smr

filtering in order to prevent information overload [16,23,26]. The remaining requirements are then refined, specified, estimated in terms of cost and resource implications, prioritized, and allocated to product releases and development units.

Flexible, scalable, and secure communication, coordination, and collaboration systems are needed to support the enactment of the governance model. Little theory-based guidance is available to help companies to design and use such systems to achieve the goals of cycle time reduction and improved product quality, service delivery, and overall effectiveness.

Design theories, unlike other theories, support the achievement of goals [27–32]. Walls *et al*. [30, p. 37] argue that the information systems (IS) 'field has now matured to the point where there is a need for theory development based on paradigms endogenous to the area itself' and call for IS design theories to fulfill that need. An IS design theory is 'a prescriptive theory based on theoretical underpinnings which says how a design process can be carried out in a way which is both effective and feasible' (ibid, p. 37). It prescribes both the design product and process aspects of a class of IS, that is, *what* are (1) the value propositions to be fulfilled by implementing an instance of the class, (2) meta-requirements describing the problem(s) to be solved by the class, (3) the meta-design prescribing the solution for the problem(s), and (4) applicable kernel theories from social and natural sciences for understanding and/or solving the problem(s) shared across all products within the class, and *how* the products should be built [30,31].

Salo and Käkölä [16] found that groupware-based requirements management systems (RMS) need to be designed and used to redesign and enact the earliest phases of product development effectively in multi-site, cross-functional organizations. They developed an IS design theory for the class of RMS in order to (1) facilitate the endogenous theory development in the context of RMS research, (2) to help RMS designers build successful RM systems for creating, prioritizing, refining, storing, and managing requirements, and (3) to guide organizations in evaluating and deploying RMS. However, the benefits afforded by RMS were hampered if the RMS instances prescribed by the IS design theory were not integrated with the systems used in the downstream phases in order to provide transparent end-to-end support throughout the product development life cycle [16]. For example, customer representatives responsible for entering requirements could not use RMS instances to follow-up if and when the requirements would be implemented, lowering their interests in entering the requirements. The scope of the IS design theory should thus be broadened to design systems that support the life cycle more comprehensively.

This research focuses on integrating requirements management with release management that is concerned with the identification, packaging, and delivery of product's elements [33]. Releases can be realized through various organizational arrangements such as release projects in organizations structured around products [34] and permanent release teams in organizations responsible for the long-term development and maintenance of strategic software and hardware assets. An illustrative scenario of release management practices for software product businesses is presented next. Each product identified during product line roadmapping is developed incrementally in release projects that follow the release plan and typically last from a few months to a year. Each release project is executed in a number of iterative cycles in which new features are added and the product quality is improved so that every cycle yields a tested and stable product version. During each cycle, feedback is collected from key stakeholders and used to plan and execute the next cycle(s). In addition to the traditional project management activities, release management determines how many cycles and internal releases are needed (for testing purposes) in a release project, refines the

DOI: 10.1002/smr

requirements identified during product line roadmapping, allocates the requirements to the most appropriate cycles, and schedules the cycles. It thus ensures that internal and external releases meet the (specified and managed subset of) requirements identified and agreed upon in the front end of product development.

Based on our extensive industrial experience and the review of academic literature, we hypothesize that the theoretical validity and practical relevance of the IS design theory for the class of RMS can be enhanced most effectively by extending the theory to provide integrated support for requirements and release management. The extant literature provides little guidance for designing and deploying integrated requirements and release management systems (RRMS). This article develops the design product theory (i.e., the product aspects of the IS design theory) for the class of RRMS (cf. [32]). It addresses the following research question:

- What are the necessary and sufficient properties for the class of RRMS in a multi-site and multi-partner environment?

The main contributions of the article are the meta-requirements of the design product theory and a meta-design that partially meets the meta-requirements. They are crystallized and validated based on (1) a case study in a global organization that deploys an RRMS instance organization-wide for effective requirements and release management and (2) a literature review in the areas of requirements management, release management, and process integration.

The design product theory for the class of RRMS can be useful and generic only, if the two key concepts *requirement management* and *release management* and the scope of the theory are clearly defined. In this article, the two concepts refer to generic requirements and release-related actions, information entities, and roles, which can be adopted throughout the multi-site and multi-partner organization regardless of (1) the organizational design, (2) the product characteristics, and (3) the selected software and/or hardware development methods. As a result, the theory is independent of issues such as: how product lines and platforms are organized, which types of customers exist, and how much the efforts and times needed to develop different types of products vary. We have determined the scope of the theory by analyzing the RRMS instance in the case organization. The instance has been successfully used for years without any major design changes whereas the case organization has instituted numerous major organizational changes. The RRMS design invariance has been possible because the organization has scoped the design effectively by (1) determining the generic requirements and the release-related actions, information entities, and roles that always need to be supported by the RRMS instance and (2) interfacing the generic design to various (A) project management practices and systems deployed to plan and monitor the project resources and costs, (B) release planning practices deploying different heuristics, methods, and systems on a case-by-case basis to plan one or more releases based on only a *limited* set of instances of information entities (e.g., some features and a limited number of releases), and (C) product portfolio management practices (where the portfolio of products is agreed). In sum, requirements and release management processes and enabling RRMS instances, respectively, need to be interfaced with project management, release planning, and product portfolio management processes and enabling systems. For example, RRMS instances need to provide middle managers responsible for requirements and release management processes with good overviews of *all* the requirements, features, and releases. During release planning a *subset* of all the features and releases is planned using various heuristics, methods, and systems. The RRMS instances serve as sources of feature and release information. Release planning usually requires information about other issues

DOI: 10.1002/smr

(e.g., available resources) from other sources too. The resulting release plans are then made available through the instances, so that all the teams working on the releases involved in a release plan can see the release schedule and the middle management can monitor the development efforts.

The design product theory has been created partially based on the experiences in the global case organization to ensure that the meta-design is flexible and scalable, that is, the RRMS instances following the meta-design can handle large volumes of information entities and their relationships (provided that the necessary hardware resources are available) and enable diverse organizational designs, development methods, and types of products (including both hardware and software). However, we have made every effort to simplify the meta-design so that even small and medium-sized organizations can leverage it to implement RRMS solutions. The design process aspects of the IS design theory for the class of RRMS are not elaborated because our industrial experiences indicate that, at least in the context of the class of RRMS, it is most useful to prescribe the design product and let the designers adopt the development methods most suitable for implementing the design product in their socio-technical contexts.

2. DESCRIPTION OF THE CASE ORGANIZATION AND THE RESEARCH METHOD

A literature review was performed to develop preliminary meta-requirements and meta-design elements before the case study started. Later on, it was complemented by a review of the commercially available RRMS. The review was essential to reduce bias induced by the single case study and ensures the generalizability of the meta-requirements and the meta-design to the maximum possible extent [35]. Potential meta-design elements that according to the review were peculiar to the organization or its RRMS instance (hereafter, the RRMS) were eliminated. For example, the RRMS consisted of numerous information entities but many of them were peculiar to it because they reflected the same underlying concepts in different abstraction levels to facilitate the technical implementation of the RRMS.

The RRMS-enabled requirements and release management process had been institutionalized across the organization by the time the study was started. Business units ran product lines in which product programs produced product releases under the guidance of product roadmaps and release plans for customers in specific market segments. Product programs deployed software and hardware platform releases developed by internal platform units, inter-organizational consortiums, and external providers. The platform releases integrated hardware and/or software component releases that were developed internally or by partners or purchased off-the-shelf from external providers. Requirements were collected from markets, service providers, and other internal and external sources. Requirements triage was then conducted to eliminate requirements that did not warrant further actions. Acceptable requirements were allocated to the appropriate units and component providers using the RRMS and iteratively refined into increasingly detailed product, platform, and/or component features that could be scheduled, implemented, and released by the individual development teams. The process typically involved complex negotiations between stakeholders.

Product lines and platform units produced a diverse set of products that were in the different stages of the product life cycles, targeted different markets, and had different component vendors. They could thus vary their RRMS-enabled requirements and release management processes within

DOI: 10.1002/smr

the boundaries agreed upon at the organizational level. For example, product, software platform, and software component releases could be planned months in advance whereas features, releases, and release dependencies of hardware platforms could be planned even one or two years in advance depending on how accurately the providers could estimate their release plans and schedules for new hardware components. Product programs that needed to develop and deliver new products quickly could utilize the RRMS (1) to know which critical new features the platforms were planning to release and when and (2) to link their requirements to the features.

The RRMS was a proprietary Lotus Domino-based application developed in the organization. An organization-wide Lotus Domino infrastructure had been institutionalized before the development started and the organization was competent in developing and deploying Domino applications. The RRMS had been productized, that is, one RRMS design and repository was used. Different organizational units had been closely involved in designing the RRMS from the beginning and had become committed to using and further developing it. They considered the RRMS highly malleable to the changing business needs partly because the organization controlled it and business units did not need to negotiate with external vendors when changes were needed.

Requirements, features, and releases were the key information entities to be managed throughout their life cycles using the RRMS. Most importantly, the RRMS was used to continuously manage dependencies within and between the information entities and enable traceability between the entities and all the organizational units responsible for creating and updating the entities during their life cycles. Requirement, feature, and release managers were responsible for managing the respective entities. Product line managers coordinated product programs and the associated platforms. For example, product line requirement managers received requirements from sources such as product marketing. If the requirements could not be addressed within the product programs because they belonged to the scope of the product platforms, product line requirement managers allocated them to the appropriate platforms and later on followed their progress using the RRMS. Release managers in the various levels used the RRMS to scope and schedule releases and to create and analyze dependencies between releases. Line managers used the RRMS to ensure the availability of the appropriate resources when needed. The RRMS also enabled release teams to search and locate reusable assets quickly, substantially increasing the perceived productivity.

However, while using the RRMS for achieving these objectives, some problems prevailed. Orchestration of complex parallel development programs, involving multiple internal and external development units across multiple sites, was challenging and coordination breakdowns sometimes occurred. For example, product releases depended on platform releases, which, in turn, could depend on other platform releases and/or component releases. Component providers ran their own businesses and product platforms were not necessarily their most important customers. Component providers sometimes had to change their commitments to meet the emerging business needs, resulting in issues such as delayed releases. The RRMS was critical for managing these dependencies and recovering from breakdowns. For example, if a release was unexpectedly delayed, the release manager updated the release schedule and the information about the delay was immediately available to all stakeholders. The schedules and scopes of the releases dependent on the delayed release could then be revised as necessary.

Indeed, the ability of the RRMS to support efficient routines and the recovery from complex breakdowns was a major reason for the successful organization-wide institutionalization of the RRMS-enabled requirements and release management process. After institutionalization, the use

DOI: 10.1002/smr

of the RRMS has been relatively stable with respect to measures such as the number of active users and the number of documents created. For example, the personnel of all product programs have entered into the RRMS the requirements for the platforms that the programs deploy. During a 3-month period in 2008, the RRMS was used in read-only mode by more than 6000 people and updated by more than 2000 people within the organization. In addition, around one hundred people used the RRMS in external service providers' sites. By the end of 2008, it contained about 22 000 active requirements, more than 50 000 active features, and 23 000 active releases. The documents that were no longer active had been automatically archived.

The RRMS was critically important for the top and middle management because it was the primary organization-wide IS containing real-time information about all the releases and associated requirements and responsible stakeholders. While there were many reasons contributing to the success of the RRMS within the organization, it is crucial, from the viewpoint of creating an IS design theory for the class of RRMS, that the RRMS-enabled real-time transparency and management of product development within the entire organization. Other significant factors contributing to the success of the RRMS were: it imposed the appropriate amount of control on the people using it, its information model included mostly textual descriptions of information and minimized redundancy of information between the RRMS and related IS, and it was easy for users to add information to it. In addition, since the RRMS focused on release management, it did not replace higher-level product portfolio management and product line roadmapping systems, which require advanced algorithmic models (e.g., what if-analyses, cost and effort estimation, optimization of inter-dependent releases) and visualization techniques. But all the results (e.g., feature priorities, allocations of features to releases, and release schedules) from using such systems had to be stored in the RRMS because (1) they guided the planning and implementation of releases and (2) the middle management based its product development decision making largely on the information available in the RRMS.

The software and hardware development processes and the supporting IS were not significantly affected either, because they were beyond the scope of the RRMS. Software development efforts increasingly leveraged agile development practices. The RRMS thus could not impose unnecessarily stringent control mechanisms on them. Only the inputs to and the deliverables of management and the software and hardware development processes and systems were dealt with by the RRMS. For example, if requirements in the RRMS needed specific product or organizational models to make them understandable to executives, managers of service providers, or other critical stakeholders, the models were crafted in appropriate modeling environments as necessary and hyperlinked or, sometimes, imported to the RRMS. Unified Modeling Language (UML) was deployed [36] but thorough UML modeling was conducted only when necessary because the organization managed tens of thousands of requirements and features through the RRMS. The priorities, schedules, and other information stored in the RRMS could be used to prioritize the modeling efforts. The models were created, modified, and managed throughout their life cycles using the modeling environments instead of the RRMS. The analysis of RRMS-enabled processes and interfacing IS has thus helped us to scope the design product theory for RRMS appropriately. More details concerning the organization and the RRMS and related IS are beyond the scope of the article.

Two of this article's authors, a doctoral student and a master's student in IS research, worked full time in the organization during a 6-month study period. The RRMS had become increasingly complex over the years when its designers had tried to meet the ongoing influx of new requirements.

DOI: 10.1002/smr

While its functionality had been documented well, the organization was keen to further develop it. A current state analysis of the RRMS was thus deemed necessary to better understand its limitations and possibilities. Major design changes were not realized during the study. The authors had access to all the relevant information and could interact with all people who had been involved with the RRMS design. They observed the use of the RRMS, analyzed documentation, discussed informally with various stakeholders, and conducted six semi-structured interviews with middle-level managers who had been involved with the design and use of the RRMS for process improvement. After interviews were completed, the interviewees were provided with interview transcripts and summaries. Interviewees reviewed the meta-requirements and proposed new ones that were added to the original set if all the interviewees considered the proposed meta-requirements critical for the class of RRMS. The proposed meta-requirements and meta-design were used in the organization for further development of the RRMS. The authors retained access to the organization and periodically observed the successful co-evolution of the RRMS and the organization until the time of completing this article.

Walls *et al*. [30,31] argue that kernel theories from natural and/or social sciences need to be identified and used to derive and govern meta-requirements of IS design theories. Interestingly, the people the authors interviewed or otherwise interacted with could not specify academic articles or theories influential during the RRMS design. They were experts with long organizational tenures and relied on theories-in-use [37] developed primarily through social interactions (cf. [38]), experiences from earlier projects, and agile development practices instead of academic articles, kernel theories, or design theories. The authors thus became increasingly intrigued with how to build such a simple but scalable and effective IS design theory for the class of RRMS based on the case study that IS designers and managers in other organizations would be willing to use the theory in addition to trial and error mechanisms and long-reinforced theories-in-use. Kernel theories were determined to be out of the scope of the design product theory because no empirical evidence could be found about their usefulness in the context of the RRMS design.

3. META-REQUIREMENTS OF THE IS DESIGN PRODUCT THEORY FOR THE CLASS OF RRMS

This section presents the meta-requirements for the design product theory for the class of RRMS. They are introduced by revising a framework of Salo and Käkölä [16]. The framework considered meta-requirements in relation to three categories of services that RMS have to offer: (1) communication, (2) control, and (3) change. Communication refers to the ability of RMS to disseminate requirements information within an organization, including information about the rationale for RM and its relationships to the external environment. Support for control ensures that requirements are dealt with in accordance with the approved principles and procedures. Support for change is needed because products, technologies, and customers change and RMS must remain amenable to adjustments at all levels of the RM activity.

The three categories are valid for the class of RRMS but two new ones are needed: (4) Platform-based product development and (5) Process integration. Platform is the physical implementation of the baseline entity that contains the common business requirements for all the derivative products

DOI: 10.1002/smr

Table I. A framework for categorizing the meta-requirements of the design product theory for the class of RRMS.

Communication	Control	Change	Platform development	Process integration
• Prioritization and valuation of requirements and the allocation of requirements into releases • Traceability • Single capture of information	• Content ownership and accountability • Management and coordination • Creating and sharing of metrics information • Access rights and information security	• Version management of requirement documents • Release re-planning • Change management and impact analysis • Defining and maintaining the requirements baseline	• Creation and reuse of reusable assets	• Process transparency • Providing high-quality information • Providing information at the right and consistent level of detail

that the platform has been designed to support (cf. [2,7,39–41]). Market-driven product development occurs on top of the platform. End-to-end process integration is necessary to ensure that all requirement and release managers have all the high-quality information they need to be available at the right level of detail when they need it. Process integration also helps to ensure that all the release level information is available in a repository so that the managers can analyze it and take decisions by means of other IS (which are beyond the scope of the design product theory presented in this article) that aggregate the requirement, feature, and release level information and link it with product portfolios and roadmaps, release plans, and other information related to strategic business management. Table I classifies all meta-requirements.

3.1. Meta-requirements in support of communication

3.1.1. Prioritization and valuation of requirements and the allocation of requirements into releases

Requirements must be allocated into releases using requirement prioritization and valuation methods that enable the most crucial requirements to be implemented and released first [42,43]. The methods are typically based on trade-off analysis between the economic values and implementation costs and resource constraints associated with the requirements [42, p. 140] and [22,44]. Moreover, all stakeholders do not have the same relative importance and each stakeholder may valuate each requirement very differently [43,45,46].

According to the interviews, customer involvement in valuation, prioritization, and selection adds value to these processes: '*Requirements can be prioritized to releases by communicating with customers and agreeing on what functionalities are wanted and on what timetable.*' RRMS instances must provide the prioritization and valuation methods with the necessary requirement, feature, and release information and store the resulting valuations, priorities, and allocations.

DOI: 10.1002/smr

3.1.2. Traceability

The purpose of requirement management is to achieve complete traceability from customers via the organizational departments to delivery [12, p. 69]. Traceability improves risk assessment, project scheduling, and change control [13, p.12]. All the interviewees agreed that traceability from requirements elicitation to product delivery is critical for RRMS success. The interviewees also suggested other needs for traceability (e.g., linking components, errors, and use cases). But it is expensive to collect and manage traceability information [47, p. 129]. Based on the analysis of the case organization, only the following traceability meta-requirements must always be implemented by RRMS instances:

- Ability to trace forward from requirement sources to requirements, from requirements to subsequent features and corresponding design elements and designs [48], and from features to future releases defined by release plans.
- Ability to trace backward from releases to the features packaged in the releases and from the features to the requirements implemented by the features [48].
- Ability to trace from requirements directly to design entities and backward [47].
- Ability to trace requirements dependencies [47,49] and release dependencies.

3.1.3. Single capture of information

RRMS instances must be the single capture points for requirements, features, and releases, ensuring that there is no redundant and inconsistent requirement, feature, and release information in the organization and that all requirements are handled appropriately in a single effective and transparent process reducing the risks of missing or forgetting requirements. RRMS instances should thus be (1) easy to use for occasional users in order to ensure that they enter the information, (2) interfaced to related systems such as requirements and architecture modeling and defect management environments, and (3) interfaced to partners' systems to ensure that partners can create, update, and review information as necessary.

3.2. Meta-requirements in support of control

3.2.1. Content ownership and accountability of experts

Requirements and release management activities should have appropriate owners who establish and reinforce appropriate norms for them [12, p.70]. Content and process ownership can be enhanced by assigning roles with clearly defined responsibilities to persons. For example, a set of requirements can be allocated to a requirement manager while a particular release can be assigned to a release manager. The role definitions and assignments improve the accountability, enable evaluations of people with respect to their role expectations (e.g., release managers may be evaluated based on the quality of releases they are responsible for), and can ensure that all agreed-upon-deliverables are delivered in time and meet the defined quality criteria. Role-based management also facilitates organizational (e.g., people may continue their work in their old roles in a new organization) and individual level changes (e.g., a new person takes responsibility of a role) [23,50]. RRMS instances must thus meet this meta-requirement to facilitate personnel evaluation, process execution and improvement, and quality control.

DOI: 10.1002/smr

3.2.2. Management and coordination

RRMS instances must enable the coordination of flows of requirements and the allocation of requirements to releases through managerial activities and decisions. For example, decisions need to be taken concerning the acceptance of particular requirements to the development process and the maturity levels of the requirements. To allocate requirements to specific releases and implementation teams and track their progress, RRMS instances must enable the assignment of different statuses to requirements. An interviewee commented that: '*It should be possible to see from the tool who is responsible for certain parts of the process*, *who makes decisions concerning those parts*, *what the timetables are*, *and what kind of documentation should be available.*'

3.2.3. Creating and sharing of metrics information

Measurement is an essential part of process management [24,50–53]. Defined and measurable objectives are needed to evaluate the current status and develop the process. Metrics information enables the comparison of process effectiveness across organizational units and similar releases over time (e.g., product line management). RRMS instances must provide a balanced set of process quality metrics (e.g., within each organizational unit) such as (1) the ratio of releases delivered in accordance with the planned release schedule to all delivered releases, (2) the ratio of releases delivered in accordance with the planned release scope (i.e., all requirements planned and assigned to the release have been realized) to all delivered releases, (3) the ratio of releases delivered in accordance with the planned work effort to all delivered releases, and (4) the ratio of cancelled releases to all releases.

3.2.4. Access rights and information security

Access rights and information security policies are crucial in high-technology companies. Products and platforms are typically designed and implemented in complex networks or consortia of companies where competitors are involved. RRMS instances must help to ensure that partners do not access each other's sensitive information. For example, multiple partners can co-operatively build a platform and use an RRMS instance to share information about it. At the same time, they may build competing products on top of the platform and the RRMS instance must not leak any confidential information related to partners' products and objectives.

3.3. Meta-requirements in support of change

3.3.1. Version management of requirement documents

Versions of individual requirements and requirements specifications need to be controlled [54, p. 268]. Change management and document version management processes must be in place to create and maintain requirement documents and their different versions. Requirement document version management is related to the general workflow management. However, one interviewee emphasized that it is most useful in the requirement development phase, and not in later phases when the documents are relatively stable. Change management and version management processes

DOI: 10.1002/smr

should be aligned, agreed upon, and enabled by the RRMS instances: '*If the change is large*, *a new requirement can be created or the old one can be changed via the change management process. In practice*, *we could use version management for small revisions. But if the changes are too large*, *change management needs to be used.*'

3.3.2. Release re-planning

When software development is market-driven [15], release planning and requirement prioritization are parts of strategic product line roadmapping [7,42]. Especially in product programs the stakeholders need to continuously create and share knowledge to deal with uncertainty and turbulent market conditions. Release re-planning is needed when, for example, the product strategy is changed. Release re-planning may be related to planning the scope, role, and timing of every product release specified in the product roadmap or to release management (i.e., re-planning the length of development, the scope of features, and the number of iteration cycles involved in a release) when, for example, several key developers leave unexpectedly [34]. Release managers may need to re-plan releases with the stakeholders on a weekly basis.

The RRMS instances prescribed by the RRMS design product theory must thus enable the relevant stakeholders to be involved in release re-planning at the right time and at low cost and provide the stakeholders with the necessary requirement, feature, and release information for re-planning both individual releases and all releases involved in a roadmap. Release managers can then use appropriate release planning heuristics, methods, and systems to plan and re-plan the releases so that the stakeholder priorities are best satisfied while the utilization of resources available to release development is maximized [46]. To facilitate the stakeholder involvement and to reduce the burden of re-planning, the release planning systems should generate alternative release plans when the stakeholder priorities, available resources, and/or requirement dependencies change; let the release managers rank the plans and take decisions together with the stakeholders; and help to store the resulting release information in the RRMS instances.

3.3.3. Change management and impact analysis

A clearly defined change management process is needed to estimate the impacts of possible changes and to control the changes made to the requirements and releases [23]. Change management and impact analysis enable organizations to be aware of the influences of requirement changes on the software and hardware components, test cases, resources, and the market situation. Change management and impact analysis are tricky when RRMS-enabled requirements and release management processes have not been institutionalized across the geographically distributed sites, projects, and partners involved, because no one has adequate visibility into the detailed activities of the projects. However, the institutionalization of these processes may dramatically improve the situation, because the RRMS instances must (1) trace requirements to the designs and development teams responsible for them, (2) trace requirements to the product releases delivering them, (3) trace requirement dependencies and release dependencies, and (4) clarify to all stakeholders designing a system in detail and in real-time which deliverables for that system are coming from which releases and when. For example, when the change of a major requirement for an important component is considered, an RRMS instance makes it relatively easy to see not only how much time and money the

DOI: 10.1002/smr

implementation of a new design would take based on the features affected, but also which other releases depend on those features. The relevant stakeholders can then be contacted to better understand the impacts and decide about the feasibility of the requirements change.

3.3.4. Defining and maintaining the requirements baseline

RRMS instances must support requirements baselining, that is, the freezing of the current agreed upon requirements. When the baseline decision has been made, subsequent requirements are treated as change requests and compared with the baseline. If the requests are accepted, they will enter product development through the change management process [23]. The requirements in a baseline that has been incorporated into a delivered release should not be changed in the subsequent releases. After all, the delivered release cannot be changed later on. Only a new release can be established to replace the delivered one as necessary. Changing the requirements associated with a release after the delivery of the release would yield to inconsistency and jeopardize the traceability. New requirements should thus be created for the subsequent releases and linked to the baselined requirements that have already been delivered.

3.4. Meta-requirements for platform development

3.4.1. Creation and reuse of reusable assets

Platforms are strategic organizational assets designed to be reusable and afford common features and predefined variation mechanisms through which mass-customized products can be created quickly and cost effectively [7]. Platforms consist of assets such as requirements, design elements, components, and user interfaces. RRMS instances should provide a comprehensive information model to describe and document the assets adequately. They should not only document platform releases at the time of creation but also help link the releases to all associated requirements, features, and releases. They should provide advanced search functionalities to help developers and other stakeholders to easily deploy the requirement, feature, and release information in order to search, retrieve, and leverage compatible assets in novel and possibly unforeseen ways. These issues are clarified by two interviewees:

Requirements management involves the identification and management of baselines and products [for strategic reuse]. For example, we could see from the [RRMS] tool that this [requirements] baseline is good for our purposes and we just have to change it from there and there in order to have a right configuration for our needs.'

'*One of the main purposes of the RRMS tool is to support reuse of both requirements and assets* (*modules*, *components*)*... Different knowledge search capabilities are essential.*'

3.5. Meta-requirements in support of process integration

3.5.1. Process transparency

RRMS instances must help to make integrated requirements and release management processes transparent, that is, the stakeholders involved should be able to understand the results of their

DOI: 10.1002/smr

RRMS-mediated actions (e.g., which product and platform releases would be affected and how, if a particular release was delayed by a month) and create knowledge for dealing with unexpected errors or coordination breakdowns as quickly and proactively as possible before expensive disruptions in routines and flaws in deliverables occur [23,50]. One interviewee stated: '*Process transparency is especially important in decision making situations. Another important situation is when someone cannot implement*, *for example*, *a requirement in a given timetable.*'

Most developers in the case organization could access (in read-only mode) all documents in the centralized RRMS database, critically improving the transparency. For example, release managers in platform units used the RRMS instance to follow the development and testing of the components their releases depended on and took corrective actions proactively, if the components were likely to be delayed. Transparency was also facilitated by using the RRMS instance to standardize the most critical information entities and the terms and forms used in them.

3.5.2. Providing high-quality information

Organizations have to be able to base their requirements and release decisions on high-quality (i.e., accurate, specific, relevant, reliable, timely, and accessible) information [55]. Transparent RRMS-mediated processes and committed, skillful people are crucial to ensure high quality. RRMS instances should also help users to identify which pieces of information are the most critical in each phase of the process, for example, by sending reminders and highlighting the required fields of respective information entities in the different phases.

3.5.3. Providing information at the right and consistent level of detail

Appropriate and consistent granularity of information facilitates decision making and eliminates extraneous activities required to decompose or summarize information [16,56]. The right level of detail depends on the situation. For example, highly mature requirements and release management processes can leverage much more detailed (quantitative) information than immature processes. RRMS instances must incorporate and represent requirement- and release-related information in consistent granularity levels (e.g., system requirements, subsystem requirements, and functional requirements) that are useful for different process contexts and roles. For example, RRMS instances may be used to generate requirements specifications for product releases. Each requirement in the generated specification should be testable by means of a small number of tests. If some requirements are not testable, they are probably represented on higher granularity levels than the testable ones and there is no consistent level of detail in the specification.

4. A META-DESIGN OF THE IS DESIGN PRODUCT THEORY FOR THE CLASS OF RRMS

This section first outlines a generic meta-design for the class of RRMS based on the analyses of interview transcripts, the RRMS instance in the case organization, and the literature review. It covers most meta-requirements specified in the previous section. The section concludes by explaining the

DOI: 10.1002/smr

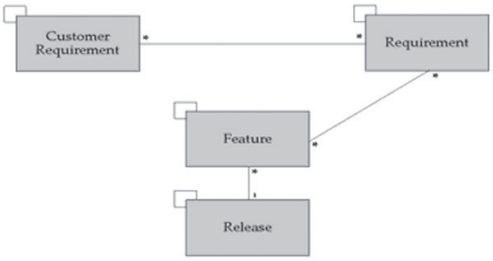

Figure 1. Information model of the meta-design of the design product theory for the class of RRMS. *Note*: The links that go back to themselves in each entity represent parent–child relationships.

linkages between the meta-requirements and the meta-design to validate the meta-design and to justify its scope.

4.1. Information model for integrated requirement management and release management process

To design an effective requirement management and release management process, the information model for the process must be defined. We have synthesized a simple but scalable model based on the literature review and a detailed examination of the RRMS instance. The experts of the case organization have reviewed and accepted the model. It consists of four entities used in the integrated process and links between and within the entities to enable the traceability, hierarchical composition (i.e., each entity can consist of any number of the same type of entities), and the appropriate granularity of information (Figure 1): Customer Requirement, Requirement, Feature and Release.

Customer Requirement is used to model requirements from the external environment. Internal and external requirements are separated to meet the meta-requirements related to platform-based product development and information security. For example, customer requirements are business critical and can provide competitive advantage by enabling organizations to focus on implementing the differentiating and high-value adding requirements. Access rights for them and for internal requirements have to be defined and enacted differently.

Requirement is used for internal requirements developed by product creation processes within an organization or a network of companies collaborating to create a joint product and/or platform. Requirement can thus be used for platform requirements related to the platform offering. In the platform context, Customer Requirements can be used as the basis for developing derivative products on top of the platform. Separation of Customer Requirement and Requirement also facilitates change management. Requirements can be changed only through negotiations with their originators. Negotiations with external requirement suppliers are often more challenging than with suppliers of internal requirements, especially when the external suppliers contribute to funding the development efforts.

Features denote the intended behaviors or properties of software-intensive systems. They are usually created and managed as hierarchical feature structures solving the problems that

DOI: 10.1002/smr

Requirements identify [57]. The solutions may reflect, for example, business processes, organizational structures, or product architectures. Feature is the largest entity in the information model containing the technical specification, workflow planning, and implementation. By using Feature as a basis of implementation and technical specification, Requirements become more manageable and there are clear traceability links to the origins of Features and to implementation phases, that is, specifications, responsible teams, and realized pieces of software code.

Product roadmaps and the associated release plans often trigger the development of new releases and provide them with high-level requirements. For every new release, a release manager is typically assigned and a Release instance is created to plan, implement, and document the resulting release. If a previous release exists, its feature set may serve as the starting point. New functional and non-functional requirements to be delivered in the release are identified and refined and Requirement and Feature instances are linked to Release instances as necessary. RRMS instances can be used to locate, reuse, and modify the existing Requirements and Features whenever possible, implement new Features as necessary, and group the implemented Features into the Release instance documenting the release. Releases can also be organized hierarchically into, for example, platform and product releases.

Customer Requirements are linked to one or more Requirements, which, in turn, are linked to one or more Features. Highest-level Requirements are typically large-scale system-level definitions of business problems. Dividing them into sub-problems (i.e., lower-level requirements) which are linked to Features enables more accurate project cost, schedule, and effort estimation and better workflow management. Features describe implementable partial solutions to the business problems. The highest-level releases are comprehensive, valuable solutions consisting of Releases and Features, whereas lower-level releases can, for example, deal with components.

4.2. Generic structures of entities

This section introduces generic structures, classes, and attributes of the entities presented in the information model. According to the design product theory, RRMS instances prescribed by the theory should include at least these entities, structures, classes, and attributes to be effective.

4.2.1. Requirement and customer requirement

Table II presents the generic structure of Requirement and Customer Requirement by revising and elaborating on the work of Salo and Käkölä [16]. Next, each class within the structure is presented.

Description class describes the intent of and justification for a Requirement and a Customer Requirement. *Version* attribute indicates the version number of the document. *Name* and *ID* attributes are used for identification and traceability.

Origin class describes where the Requirement comes from and when. For Customer Requirements the sources are external organizations. For Requirements, sources are Customer Requirements and internal organizational units.

Categorization class describes the parts (i.e., *platform*, *product*, and *responsible person*) of the product and the development organization managing the Requirement or Customer Requirement. Platform works as the base architecture for derivative products. Requirements and Customer

DOI: 10.1002/smr

Table II. Generic structure of Requirement and Customer Requirement.

Class	Question	Attributes
Description	What is the requirement about?	Name ID Description Rationale Version
Origin	Where does the requirement come from?	Author Source Date of creation
Categorization	What parts of the product and the development organization is the requirement related to?	Platform Product Responsible Person
Analysis	What are the implications of the requirement?	Status Priority Customer need Deadline for the customer need Customer value Risks Required work effort Total cost
Workflow	What should be done to this requirement next? By whom?	Allocation to Requirements/Features
History	What has been done to the requirement? When?	Assignment to Requirement/Feature responsible Information about all prior edits, editors, and changes

Requirements will be linked to the appropriate platforms and final products via Features and Releases. Responsible persons update Requirements and Customer Requirements as necessary.

Analysis class is used to probe the implications of the Requirement. *Priority* and *customer value* can be handled as one attribute, but organizations needing sophisticated valuation processes should divide them into two. *Customer need* attribute is used to describe the detailed business case, which the Requirement or Customer Requirement is trying to solve. If there is a firm deadline by which the Requirement needs to be implemented and released for use of customer(s) (e.g., in their products), the deadline must be made explicit through the *deadline* attribute. The *total cost* and *required work effort* need to be estimated [51,53] to determine whether the Requirement is feasible from the economic, personnel, and schedule viewpoints. *Risks* associated with the (Customer) Requirement need to be assessed. *Status* attribute models the requirements life cycle. Examples of requirement statuses include: New—Categorized—Analyzed—For Review—Approved/Rejected/Postponed [16].

Workflow class describes what should be done next to Requirement or Customer Requirement and by whom. Customer Requirements and Requirements are allocated, respectively, to Requirements and Features and responsible persons are assigned.

History class is used to provide information about all prior changes and editors of requirements documents [16]. It enables traceability and the development of organizational memory that is

DOI: 10.1002/smr

especially useful when routines break down unexpectedly and the reasons for the breakdowns must be found and eliminated to continue the effective execution of routines [50].

4.2.2. Feature

Each class within the Feature-entity (Table III) is presented in this section.

Description class tells the intent of and justification for the Feature. *Origin class* indicates the author, date of creation, and Requirements, if any, from which the Feature is allocated.

Categorization class links Features to products and/or product platforms and identifies the person having the feature responsibility. Because Feature is an entity for managing detailed implementation, it has a *traceability links* attribute containing links to technical specifications, documentations and code. Features tend to have complex dependencies [57]. For example, a Feature may be incorporated into a Release but its parent Feature may not be incorporated if the scope of the parent Feature is too wide for implementation in any single release. However, the parent Feature may fulfill a particular Requirement. A dependency link between these two Features thus provides valuable information for decision makers.

Analysis class contains most attributes that Requirement and Customer Requirement have, with the exception of *customer value*-attribute used to decide whether Requirements or Customer Requirements should be implemented or not. The *required work effort* needs to be estimated to

Table III. Generic structure of Feature.

Class	Question	Attributes
Description	What is the feature about?	Name ID Description Rationale Version
Origin	Where does the feature come from?	Author Source Requirements Date of creation
Categorization	What parts of the product and the development organization is the feature related to?	Platform Product Responsible Person Traceability links (e.g., documentation, code)
Analysis	What are the implications of the feature?	Status Priority Customer need Risks Required work effort Realized work effort
Workflow	What should be done to this feature next? By whom?	Task description Assignment to teams/persons Assignment to Release Date when Feature is ready for Release
History	What has been done to the feature? When?	Information about all prior edits, editors, and changes

DOI: 10.1002/smr

Table IV. Generic structure of Release.

Class	Question	Attributes
Description	What is the release about?	Name ID Description Version
Origin	Where does the release come from?	Author Source Features Source Releases Date of creation
Categorization	What parts of the product and the development organization is the release related to?	Platform Product Responsible Person
Analysis	What are the implications of the release?	Status Priority Required work effort Realized work effort
Workflow	What should be done to this release next? By whom? What should be done to dependent releases?	Planned release date Actual release date Dependent Releases
History	What has been done to the release? When?	Information about all prior edits, editors, and changes

assess implementation costs and help teams in their work allocation and scheduling. It is also useful to determine *Realized work effort* when the feature is ready for release because estimation practices can be systematically improved by comparing the original work effort estimates to the actually realized work efforts.

Workflow class consists of detailed *task descriptions* together with traceability links to provide the guidelines for implementation work and to enable organizational learning through, for example, post-mortem analysis (i.e., what was planned vs realized). Before starting the work, Features are assigned to responsible teams or persons. *History* class is used to provide information about all prior changes and editors of feature documents.

4.2.3. Release

Classes within the Release-entity (Table IV) that need elaboration are explained in this section.

Description class describes what the Release is about. For example, the Release may fix some specific quality problems of the previous Release without providing new functionality. In *Origin class*, *source features* and *source releases* attributes indicate which Features and Releases are included in a Release. In *Categorization class*, a Release is related to specific product *platforms* and/or *products* and has a *responsible person*.

DOI: 10.1002/smr

Analysis describes the life cycle of a Release through *status* attribute. Statuses include: Planned—Ready to be Released (i.e., all Features belonging to the Release have been implemented, tested, and found to be stable and all lower-level Releases the Release depends on have been released)—Released. It should be noted that a Release can also be canceled but only if no other Release is dependent on it. Dependent Releases must thus be canceled or made independent from the Release to be canceled. The allocations of Features to a canceled Release must be removed in respective Feature-entities.

For every source Feature, the *required work effort* should have been estimated when the source Features were analyzed. When a Release has been delivered, it is useful to assess and document (1) the work effort that was necessary to realize each Feature, (2) the total work effort realized to implement the Release, and (3) the reasons for any major discrepancies between the estimated and realized efforts. Impact analysis practices and the accuracy of release schedules can then be improved in the future.

Because Releases constitute manageable and releasable entities, the only *Workflow-* related attributes tell the *planned* and *actual release dates* and provide links to all Releases depending on this Release. The release managers of *dependent releases* can thus be notified, for example, when the Release has been delivered or when it will be unexpectedly delayed. This information together with the information stored in the *realized work effort* attribute and *History* class is adequate for organizational learning and performance monitoring.

4.3. Validating and scoping the meta-design by analyzing how it meets the meta-requirements

This section analyzes how the meta-design satisfies the meta-requirements because 'a design artifact is complete and effective when it satisfies the requirements and constraints of the problem it was meant to solve' [27, p. 85].

4.3.1. Prioritization and valuation of requirements and the allocation of requirements into releases

Prioritization and valuation of requirements are enabled by the entities Requirement and Customer Requirement. Their attributes *priority* and *customer value* are used to store and access the prioritization and valuation information in the RRMS instances. The prioritization and valuation methods are not included in the meta-design for two reasons. First, the literature provides hardly any methods that are generalizable and scalable to meet the needs of complex industrial environments where multiple interdependent releases of interdependent products and platforms are planned simultaneously [58]. Second, the product programs of the case organization used different prioritization methods and tools because the programs differed in size, duration, and product maturity. Organizations must decide which prioritization and valuation methods they wish to use. The meta-design ensures that RRMS instances can provide the methods with most if not all the necessary information and store and share the results organization-wide.

Allocation of requirements into releases is enabled transitively through features, that is, requirements and customer requirements are allocated to features, which are linked to releases. Releases provide implemented functionality and are thus linked to features directly.

DOI: 10.1002/smr

4.3.2. *Traceability*

The information model enables bi-directional traceability between entities through *Origin* and *Workflow* classes. In Customer Requirement and Requirement, *source* attribute is used for backward traceability and *allocation to features* attribute enables forward traceability to features. *Source requirement* and *assignment to release* attributes of Feature enable, respectively, backward and forward traceability from features. *Traceability links* attribute enables the traceability from Features to implementation specific documentation and software code.

4.3.3. *Single capture of information*

Based on the analysis of the RRMS instance in the case organization, the information model is comprehensive enough so that the RRMS instances prescribed by the meta-design can be the single capture points for requirements, features, and releases in organizations.

4.3.4. *Content ownership and accountability of experts*

Content ownership and accountability are determined through the *responsible person* attribute of *Categorization* class. For example, each release has to specify who is responsible for planning, which features are released in which release. The meta-design does not detail the metrics that could be used for the measuring performance. However, it can be used as a basis for sophisticated measurement systems.

4.3.5. *Management and coordination*

The meta-design supports management and coordination across multiple, interdependent product, platform, and component releases, for example, by explicating the schedules imposed on various entities, the products and organizational units the entities are related to, and the workflows the entities are subjected to.

4.3.6. *Creating and sharing of metrics information*

The meta-design affords a balanced set of process quality metrics. Releases contain information about planned and actual release dates, making it easy to measure (e.g., within an organizational unit) issues such as what the ratio of releases delivered in accordance with the planned release schedule to all delivered releases is. Status information is readily available, making it easy to see, for example, what the ratio of cancelled releases to all releases is. Detailed work effort information can be collected, making it possible to determine, for example, what the ratio of releases delivered in accordance with the estimated work effort to all delivered releases is. It is also possible to measure the volatility of release scopes because for each release the associated Release instance documents (primarily through the *source feature* attribute) the evolution of the feature set associated with the release from the time the release is planned to the time the release is delivered or cancelled.

DOI: 10.1002/smr

4.3.7. *Version management*

Description and *History* classes enable the version management of requirements, releases, and other entities by, respectively, numbering versions and showing the actors involved with each version and the actions taken.

4.3.8. *Release re-planning*

The individuals responsible for particular features and releases have to re-plan the releases when some unexpected (coordination) breakdowns occur (e.g., Features belonging to a Release cannot be released because their implementations are unexpectedly delayed; a platform cannot be released because it depends on a component Release that has been canceled; a competitor releases a competitive product unexpectedly and fast response is necessary). Bidirectional traceability links between Features and Releases (stored in *assignment to release* and *source features* attributes) and between Releases (stored in *source releases* and *dependent releases* attributes) facilitate the implementation of the meta-requirement.

When there are numerous interdependencies between releases, between features, and between features and releases, the appropriate data stored in an RRMS instance can be transferred into a release re-planning and optimization system (cf. [42,43]) for analysis and creation of a new release plan. Prescribing the features of such systems is beyond the scope of the design product theory for the class of RRMS presented in this article because the systems are algorithmically complex, enable cost, effort, and schedule estimation based on historical data [24] and operate on a higher level of analysis than the RRMS instances where strategic and operational decisions (e.g., about the common features within and across the product lines) are taken based on information in the RRMS instance and other systems. Future research is needed to study whether it is beneficial and feasible to extend the IS design product theory for the class of RRMS so it covers such classes of systems.

4.3.9. *Change management and impact analysis*

Change management is facilitated by the *History* class in all entities. Change requests can be considered as normal (Customer) Requirements, analyzed, linked to the respective existing Requirements in the RRMS instance that are within the scope of the change, and implemented and released by following the integrated requirements and release management process. Impact analysis is enabled by *Categorization* and *Analysis* classes. *Platform* and *product* attributes show the organizational entities affected by each Requirement and Release. *Customer value* and *required work effort* attributes are used to decide the feasibility of implementing a (Customer) Requirement.

4.3.10. *Creation and reuse of reusable assets*

The hierarchical composition of Requirement, Feature, and Release entities provided by the information model enables the comprehensive documentation of product and platform releases and all associated assets. This information together with bidirectional traceability links between the entities help organizations analyze their asset base and establish, find, and use reusable assets.

DOI: 10.1002/smr

5. CONCLUSIONS AND FUTURE RESEARCH

This article has focused on the RRMS-enabled multi-site and platform-based product development. It has synthesized the meta-requirements and a meta-design of a comprehensive IS design product theory for the class of RRMS to help practitioners in both small and large software and software-intensive organizations to implement and evaluate scalable RRMS solutions. For example, organizations can use the theory (1) to ensure that the key roles (e.g., requirement manager and release manager) are established and adequately staffed in all release efforts and made responsible for the information entities specified by the meta-design of the theory, (2) to develop requirements management and release management tools and/or select and acquire commercial off-the-shelf tools, and (3) to integrate the tools into RRMS instances that meet the meta-requirements and support the information model specified in the meta-design. The validity of the theory has been enhanced by using methods such as the analysis of the RRMS instance in a case organization and by explicating the meta-requirements met by the meta-design. The meta-design is scalable because its most essential elements have been abstracted from the RRMS instance that, at the time of writing this article, has thousands of users and manages tens of thousands of documents in the case organization alone.

Owing to space limitations, the design product effectiveness hypotheses of the theory (clarifying the expected organizational benefits from using an RRMS instance (i.e., the design product) derived from the class of RRMS) are beyond the scope of this article. The hypotheses are needed for the empirical validation and possible revision of the theory in future research. The deployment of RRMS instances can be hypothesized: to reduce the resources needed in product development (e.g., through strategic reuse of product platforms and components); shorten time-to-market (e.g., through reuse and by ensuring that right information is available at the right time for the right people); improve customer satisfaction (e.g., by ensuring that requirements are transformed efficiently to product features); and improve the process and product quality (e.g., by improving the synchronization of work across multiple sites, projects, and partners; minimizing the number of errors during development; and easing up error tracking). Future research is necessary to assess, extend, and elaborate these design product effectiveness hypotheses.

Future research is also necessary to devise extensions to the design product theory such as improved RRMS support for (1) strategic product line roadmapping and release planning processes [7,34] that take a long-term view and thus steer release management and (2) finding and reusing implementation level assets that meet the needs of releases. The first extension would require future research concerning how Release entities can be used to enable general managers, product managers, and release managers to understand even better which individual product and platform releases are linked with which release plans and product roadmaps and why they are linked. The second extension would require at least the inclusion of a Component entity in the information model of the meta-design. Our preliminary industrial experiences show that Component is useful especially if it describes how and when Component instances have been tested.

The single case study methodology may not provide a sound basis for generalization [35]. Therefore, new case studies and action research projects are necessary to make the design product theory more credible for IS designers and researchers. Design science research leveraging the methods of action research [27] helps to examine the applicability of the theory by finding out to what extent organizations that want to acquire or design and implement RRMS systems can utilize the theory for those purposes. The theory can then be revised as necessary.

DOI: 10.1002/smr

ACKNOWLEDGEMENTS

The authors wish to thank Kirk Kandt from the Jet Propulsion Laboratory, Rick Kazman from the University of Hawaii, and the three anonymous reviewers for their excellent comments on the earlier versions of this article. The authors also gratefully acknowledge the support of the numerous employees of the case organization. Dr Timo Käkölä is grateful to the Academy of Finland, whose generous grant facilitated the writing of this article while he served as the Senior Research Scholar of the Academy of Finland in Claremont Graduate University, California, U.S.A.

REFERENCES

1. Brown SL, Eisenhardt KM. Product development: Past research, present findings, and future directions. *Academy of Management Review* 1995; **20**(2):343–378.
2. Meyer MH, Selinger R. Product platforms in software development. *Sloan Management Review* 1998; **40**(1): 61–74.
3. Carmel E, Agarwal R. Tactical approaches for alleviating distance in global software development. *IEEE Software* 2001; **18**(2):22–29.
4. Herbsleb JD, Moitra D. Guest editors' introduction: Global software development. *IEEE Software* 2001; **18**(2): 16–20.
5. Ramasubbu N, Krishnan MS, Kompalli P. Leveraging global resources: A process maturity framework for managing distributed development. *IEEE Software* 2005; **22**(3):80–86.
6. Käkölä T, Dueñas J (eds.). *Software Product Lines*: *Research Issues in Engineering and Management*. Springer: Berlin, 2006.
7. Pohl K, Böckle G, Van der Linden F. *Software Product Line Engineering*: *Foundations*, *Principles*, *and Techniques*. Springer: Berlin, 2005.
8. Van der Linden F, Schmid K, Rommes E. *Software Product Lines in Action*: *The Best Industrial Practice in Product Line Engineering*. Springer: Berlin, 2007.
9. Carmel E, Agarwal R. The maturation of offshore sourcing of information technology work. *MIS Quarterly Executive* 2002; **1**(2):65–78.
10. Käkölä T. Best practices for international eSourcing of software products and services. *Proceedings of 41st Hawaii International Conference on Systems Sciences* (*HICSS-41*). IEEE: New York NY, 2008.
11. Adelson B, Soloway E. The role of domain experience in software design. *IEEE Transactions on Software Engineering* 1985; **11**(11):1351–1360.
12. Grynberg A, Goldin L. Product management in Telecom Industry—Using requirements management process. *Proceedings IEEE International Conference on Software*: *Science*, *Technology and Engineering* (*SwSTE'03*), Los Alamitos, CA, U.S.A. IEEE Computer Society, 2003.
13. Halbleib H. Requirements management. *Information Systems Management* 2004; **21**(1):8–14.
14. Hrones JA Jr, Jedrey BC Jr, Zaaf D. Defining global requirements with distributed QFD. *Digital Technical Journal* 1992; **5**(4):36–46. Available at: http://www.hpl.hp.com/hpjournal/dtj/vol5num4/vol5num4art3.pdf [15 January 2010].
15. Regnell B, Höst M, Nattoch Dag J, Beremark P, Hjelm T. An industrial case study on distributed prioritization in market-driven requirement engineering for packaged software. *Requirements Engineering* 2001; **6**(1):51–62.
16. Salo A, Käkölä T. Groupware support for requirements management in new product development. *Journal of Organizational Computing and Electronic Commerce* 2005; **15**(4):253–284.
17. Akao Y. An introduction to quality function deployment. *Quality Function Deployment*: *Integrating Customer Requirements into Product Design*, Akao Y (ed.). Productivity Press, 1990; 3–24.
18. Burchill G, Fine CH. Time versus market orientation in product concept development: Empirically-based theory generation. *Management Science* 1997; **43**(4):465–478.
19. Griffin AJ, Hauser JR. Patterns of communication among marketing, engineering and manufacturing—A comparison between two new product teams. *Management Science* 1992; **38**(3):360–372.
20. Curtis B, Krasner H, Iscoe N. A field study of the software design process for large scale systems. *Communications of the ACM* 1988; **31**(11):1268–1287.
21. Ciborra C. The platform organization: Recombining strategies, structures, and surprises. *Organization Science* 1996; **7**(2):103–118.
22. Wesselius J. Strategic scenario-based valuation of product line roadmaps. *Software Product Lines*: *Research Issues in Engineering and Management*, Käkölä T, Dueñas JC (eds.). Springer: Berlin, 2006; 53–89.
23. Käkölä T, Taalas A. Validating the information systems design theory for dual information systems. *Proceedings of the 29th International Conference on Information Systems* (*ICIS*). Association for Information Systems: Paris, 2008. Available at: http://www.aisnet.org [15 January 2010].

DOI: 10.1002/smr

24. Forselius P, Käkölä T. An information systems design product theory for software project estimation and measurement systems. *Proceedings of 42nd Hawaii International Conference on Systems Sciences* (*HICSS-42*). IEEE: New York NY, 2009.
25. ISO. Information technology—Software measurement—Functional size measurement—Part 6. *ISO/IEC 14143*, International Organization for Standardization, 2006. Available at: http://www.iso.org [15 January 2010].
26. Van De Ven AH. Central problems in the management of innovation. *Management Science* 1986; **32**(5):590–607.
27. Hevner AR, March ST, Park J, Ram S. Design science in information systems research. *MIS Quarterly* 2004; **28**(1):75–105.
28. Markus ML, Majchrzak A, Gasser L. A design theory for systems that support emergent knowledge processes. *MIS Quarterly* 2002; **26**(3):179–212.
29. Van Aken JE. Management research based on the paradigm of the design sciences: The quest for field-tested and grounded technological rules. *Journal of Management Studies* 2004; **41**(2):219–246.
30. Walls JG, Widmeyer GR, El Sawy O. Building an information system design theory for vigilant EIS. *Information Systems Research* 1992; **3**(1):36–59.
31. Walls JG, Widmeyer GR, El Sawy O. Assessing information system design theory in perspective: How useful was our 1992 initial rendition? *Journal of Information Technology Theory and Application* 2004; **6**(2):44–58.
32. Walsh KR, Dickey MH. Structured modeling group support systems: A product design theory. *Information and Management* 2003; **41**(5):655–667.
33. Software engineering—Guide to the software engineering body of knowledge (SWEBOK). *ISO. ISO/IEC TR 19759*, International Organization for Standardization, 2005. Available at: http://www.iso.org [15 January 2010].
34. Rautiainen K, Lassenius C, Vähäniitty J, Pyhäjärvi M, Vanhanen J. A tentative framework for managing software product development in small companies. *Proceedings of 35th Hawaii International Conference on System Sciences* (*HICSS-35*), 2002; 3409–3417.
35. Yin RK. *Case Study Research*: *Design and Methods* (3rd edn). Sage Publications: Beverley Hills CA, 2003.
36. Koivulahti-Ojala M, Käkölä T. Framework for evaluating the version management capabilities of a class of UML modeling tools from the viewpoint of multi-site, multi-partner product line organizations. *Proceedings of the 43rd Hawaii International Conference on Systems Sciences* (*HICSS-43*). IEEE: New York NY, 2010.
37. Argyris C, Schon DA. *Organizational Learning II*: *Theory*, *Method*, *and Practice*. Prentice-Hall: Englewood Cliffs NJ, 1995.
38. Perry DE, Staudenmayer NA, Votta LG. People, organizations, and process improvement. *IEEE Software* 1994; **11**(4): 36–45.
39. Bosch J. Maturity and evolution in software product lines: Approaches, artefacts and organization. *Proceedings of the Second Software Product Line Conference* (*SPLC2*) (*Lecture Notes in Computer Science*). Springer: Berlin, 2002; 257–271.
40. McGrath ME. *Product Strategy for High-Technology Companies*: *How to Achieve Growth*, *Competitive Advantage*, *and Increased Profits* (2nd edn). McGraw-Hill: New York NY, 2001.
41. Meyer MH, Lopez L. Technology strategy in software products company. *The Journal of Product Innovation Management* 1995; **12**(4):294–306.
42. Carlshamre P. Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering* 2002; **7**(3):139–151.
43. Ruhe G, Saliu MO. The art and science of software release planning. *IEEE Software* 2005; **22**(6):47–53.
44. Gilb T. *Principles of Software Engineering Management*. Addison-Wesley: Reading MA, 1998.
45. Greer D, Ruhe G. Software release planning: An evolutionary and iterative approach. *Information and Software Technology* 2004; **46**(4):243–253.
46. Momoh J, Ruhe G. Release planning process improvement—An industrial case study. *Software Process*: *Improvement and Practice* 2006; **11**(3):295–307.
47. Kotonya G, Sommerville I. *Requirement Engineering*: *Processes and Techniques*. Wiley: England, 1998.
48. Davis A. *Software Requirements*: *Objects*, *Functions*, *and States*. Prentice-Hall: Eaglewood-Cliffs NJ, 1992.
49. Forsgren P, Daugulis A. Requirements engineering in Control Center Procurement Projects: Practical experiences from the power industry. *Proceedings of the 3rd International Conference on Requirements Engineering*. IEEE: New York NY, 1998.
50. Käkölä T, Koota K. Dual information systems: Supporting organizational working and learning by making organizational memory transparent. *Journal of Organizational Computing and Electronic Commerce* 1999; **9**(2 and 3):205–232.
51. Bundschuh M, Dekkers C. *The IT Measurement Compendium*: *Estimating and Benchmarking Success with Functional Size Measurement*. Springer: Berlin, 2008.
52. Daskalantonakis MK. A practical view of software measurement and implementation experiences within Motorola. *IEEE Transactions on Software Engineering* 1992; **18**(11):998–1010.
53. Jones C. *Applied Software Measurement* (3rd edn). McGraw-Hill: New York NY, 2008.
54. Wiegers KE. *Software Requirements*: *Practical Techniques for Gathering and Managing Requirements* (2nd edn). Microsoft Press: Washington, 2003.

DOI: 10.1002/smr

55. O'Reilly CA. Variations in decision makers' use of information sources: The impact of quality and accessibility of information. *Academy of Management Journal* 1982; **25**(4):756–771.
56. Aubert BA, Vandenbosch B, Mignerat M. Towards the measurement of process integration. *Proceedings of the Annual Conference of the Administrative Sciences Association of Canada*, Halifax, NS, 2003.
57. Zhang W, Mei H, Zhao H. Feature-driven requirement dependency analysis and high-level software design. *Requirements Engineering* 2006; **11**(3):205–220.
58. Lehtola L, Kauppinen M. Suitability of requirements prioritization methods for market-driven software product development. *Software Process Improvement and Practice* 2006; **11**(1):7–19.

AUTHORS' BIOGRAPHIES

Timo K. Käkölä is a tenured associate professor of information systems and software business research in the Department of Computer Science and Information Systems, University of Jyväskylä, Finland. He has PhD, Ph. Licentiate., and MSc degrees in Computer Science and Information Systems Science from the University of Turku, Finland and an MBA in management from the Helsinki School of Economics and Business Administration, Finland. His research interests include IT-enabled organizational designs for effective organizational and inter-organizational creation and sharing of knowledge, the design of information systems architectures that support the learning, enactment, breakdown management, and redesign of knowledge-intensive work processes, and the business models and processes of software-intensive product-oriented ventures. He has published a number of articles in leading scientific journals and conferences such as Journal of Management Information Systems, Journal of Organizational Computing and Electronic Commerce, International Conference on Information Systems, and Hawaii International Conference on Systems Sciences.

Mervi Koivulahti-Ojala received her MSc degree in Information Systems Science from the University of Turku. She has over ten years of working experience in the IT industry as an R&D specialist and line manager. Her doctoral research in the University of Jyväskylä focuses on the design, evaluation, and acquisition of computer-based information systems for supporting the R&D efforts of large global organizations in the areas of requirements engineering, UML modeling, and systems analysis and design.

Jani Liimatainen received his MSc degree in Information Systems Science from the University of Jyväskylä. He is currently leading the Finnish Product Innovation practice within Accenture Management Consulting. He focuses on strategic product management, R&D effectiveness, and product portfolio management in several industrial domains.

DOI: 10.1002/smr

II

FRAMEWORK FOR EVALUATING THE VERSION MANAGEMENT CAPABILITIES OF A CLASS OF UML MODELING TOOLS FROM THE VIEWPOINT OF MULTI-SITE, MULTI-PARTNER PRODUCT LINE ORGANIZATIONS

by

Mervi Koivulahti-Ojala & Timo Käkölä, 2009

Proceedings of the 43rd Hawaii International Conference on Systems Sciences

.

Framework for Evaluating the Version Management Capabilities of a Class of UML Modeling Tools from the Viewpoint of Multi-site, Multi-partner Product Line Organizations

Mervi Koivulahti-Ojala & Timo Käkölä
University of Jyväskylä
40014 University of Jyväskylä, Finland
{meelheko, timokk}@jyu.fi

Abstract: UML models are widely used in software product line engineering for activities such as modeling the software product line reference architecture, detailed design, and automation of software code generation and testing. But in high-tech companies, modeling activities are typically distributed across multiple sites and involve multiple partners in different countries, thus complicating model management. Today's UML modeling tools support sophisticated version management for managing parallel and distributed modeling. However, the literature does not provide a comprehensive set of industrial-level criteria to evaluate the version management capabilities of UML tools. This article's contribution is a framework for evaluating the version management features of UML modeling tools for multi-site, multi-partner software product line organizations.

Keywords: Global software development, modeling tool, software product line organization, tool evaluation, UML modeling, version management

1. INTRODUCTION

To succeed in the global markets of software-intensive products, high-tech companies need to shorten the cycle time of new product development while improving product quality and service delivery along with maintenance or reduction of the total resources required [6;20;25]. This concern can be dealt through internal or external strategies. Internal strategies include global software development, where development resources are distributed globally to reap cost benefits, leverage specialized competencies, and address specific needs of geographically-defined markets [7;13;31], and software product line engineering and management, that is, the strategic acquisition, creation, and reuse of software assets [19;24;30]. External strategies include acquiring commercial off-the-shelf components and outsourcing software development, maintenance, and related services to best-in-class service providers [8;18].

This paper focuses on the software product line engineering strategy in the context of global software development. Software product line engineering is an industrially validated methodology for developing software products and software-intensive systems and services faster, at lower costs, and with better quality and higher end-user satisfaction. It differs from single system development in two primary ways [30]:

1. It needs two distinct development processes: domain engineering and application engineering. Domain engineering defines the commonality and variability of the software product line, thus establishing the common software platform for developing high-quality applications rapidly within the line. Application engineering derives specific applications by strategically reusing the platform and by exploiting the variability built into the platform.
2. It needs to explicitly define and manage variability. During domain engineering, variability is introduced in all software product line assets such as domain requirements, architectural models, components, and test cases. It is exploited during application engineering to derive applications mass-customized to the needs of different customers and markets.

Software product line engineering involves higher levels of abstraction than single-system development methods because the platforms require substantial investments, have long life cycles, and have to be generally applicable to a wide range of products. Without appropriate abstractions, such platforms cannot be built and variability cannot be managed effectively. Industrially validated modeling approaches and commercially available modeling tools are critically important to deal with the abstractions. In addition to traditional system modeling, variability modeling is required in product line engineering to explicitly document how the applications within the product line can vary.

To model the variability of a product line, two approaches have been proposed in the literature. The first, traditional approach has been to integrate variability modeling in the systems modeling language such as Unified Modeling Language™ (UML) [12;27] by appropriately extending the metamodel of the language [4]. The second approach, orthogonal variability modeling, distinguishes between a variability model and a system model [30]. Orthogonal variability models are easier to apply in practice and scale better than integrated variability models. They usually describe the variability using a graphical notation. One reason, orthogonal variability modeling is not yet extensively used in the industry, is that there are no commercially available modeling tools to support it. Therefore, this paper will focus on the traditional integrated modeling approach.

System models can be applied, for example, to model static and dynamic aspects of the software product line reference architecture, to conduct detailed domain and

978-0-7695-3869-3/10 $26.00

application design, and to automate software code generation and testing. UML has become the most widely accepted software system modeling language [4]. It can also be used to model embedded, business, and real-time systems.

UML modeling tools supporting sophisticated version management are critical for managing parallel and geographically distributed modeling activities. However, the extant literature does not provide a comprehensive set of industrial-level criteria to evaluate the version management capabilities of UML tools. If modeling tools fail to support version management in multi-site, multi-partner development environments, the modeling process may be ineffective and modeling tools may not be used optimally. Ineffective tool deployment is expensive since there will be substantial costs without realizing the potential benefits.

The main contribution of this paper is a framework consisting of a set of industrial-level criteria for evaluating UML modeling tools. The framework can be used in practice to determine whether particular UML tool instances support collaborative modeling through version management in multi-site and multi-partner product line organizations. The framework has been created based on a literature review and empirical experiences of the first author during a tool evaluation project. The goals of the project for a large global product line organization, which leveraged multi-site, multi-partner practices, were to identify and evaluate commercial UML tools and to select one for global deployment.

The paper is organized as follows. In Section 2, the basic concepts related to UML modeling and modeling tools are introduced. In Section 3, the existing research related to version management capabilities afforded by UML modeling tools is evaluated. In Section 4, the research method and the case organization are described. In Section 5, the role of version management in multi-site, multi-partner product line organizations is discussed and the framework consisting of a set of evaluation criteria is proposed. In Section 6, two commercial UML modeling tools are evaluated using the framework. In Section 7, the validity and usefulness of the framework are evaluated. Section 8 concludes the paper.

2. FUNDAMENTALS OF MODELS AND UML MODELING TOOLS

In this section, the notion of a model is explained and a framework (Table 1) is created to depict how models can be applied for different types of communication in the context of product line engineering. The role of modeling tools in supporting the shared creation and maintenance of models is then discussed.

2.1 A framework for analyzing product line models

The interpretation of models involves the assignment of meanings to the symbols and truth-values to the sentences of the models [33, p.74]. Models can be used for sharing information between humans, between machines, and between humans and machines.

Counterparts in communication	*Example in product line modeling*	*Example reference related to product line modeling*
Human to human	Modeling requirements Modeling the software product line reference architecture	Product line variability modeling with UML 2.0 [4] Software Product Line Engineering with the UML: Deriving Products [35]
Human to machine	Test automation	Product Line Use Cases: Scenario-Based Specification and Testing of Requirements [5]
Machine to human	Reverse engineering	Feature-oriented Re-engineering of Legacy Systems into Product Line Assets – a Case Study [15]
Machine to machine	Model transformations Code generation	Code Generation to Support Static and Dynamic Composition of Software Product Lines [34]

Table 1. A framework for analyzing product line models as means of communication and information sharing.

Human to human communication

Modeling language independent and dependent modeling approaches have been proposed to support human communication. Kruchten [17] argues that software architectures should be depicted from five modeling language and tool independent viewpoints: logical, process, physical, development, and use case. The depictions allow for the separation of the concerns of the various architectural stakeholders (e.g., end-users, developers, systems engineers, and project managers). In the area of software product lines, the modeling languages need to enable the modeling of commonalities and variabilities. For this purpose, Bayer et al. [4] present a consolidated variability meta-model with a unified terminology and representation that enables variability specification during domain engineering and variability resolution during application engineering. The model helps stakeholders to collaborate throughout the life cycles of software product lines and vendors to develop interoperable commercial and open-source modeling tools.

Human to machine communication

Models can be used (primarily during requirements engineering) to codify human knowledge and organizational rules and resources into forms that enable computerized actions. The Object Constraint Language (OCL), being part of the UML standard, is useful in product line engineering for (1) defining rules to which domain model elements must conform, so application models can be derived from the domain models, and for (2) validating the application models that reuse and possibly modify the domain models. Models are also crucial for

validating the codified knowledge. For example, Leppänen [22] has used formal methods for testing models. In the area of product line engineering, models have been used to test application requirements derived from domain requirements [5]. The models have also been used to derive application test cases from reusable domain test cases, which have been created to verify and validate domain requirements [32].

Machine to human communication

Humans routinely use computer-based information systems for decision-making and analysis. For example, reverse engineering tools are used to automatically create architectural and other models based on code. This is especially useful in the context of open source software that is seldom accompanied by detailed design models [2]. The capabilities of reverse engineering tools to generate product line models with explicitly defined variability from application code are limited partly because much of the variability has typically been resolved by the time the code has been created. For example, if an application has been derived that contains no optional features afforded by the product line, the code related to the optional features may be entirely missing from the application code, making it impossible to determine based on the code which optional features may have been available in the product line. To our knowledge there are no reverse engineering tools, which could interpret the code and transform the implemented variable and configurable elements of software product lines into variability models.

Machine to machine communication

Models can be automatically transformed into other models or to code. Model Driven Architecture (MDA) is a framework based on the UML and other industry standards that promote the creation of machine-readable, abstract models [16]. The models are developed independently of the technology platforms; stored in and shared through standardized repositories; and automatically transformed into database schemas, software code, and other assets for various platforms.

Several modeling tools support platform-independent modeling, code generation, XML, and/or database schema generation features. For example, when a product line consists of similar products running on different operating systems, domain engineering can leverage platform-independent modeling to design and implement the common parts of the product line for the operating systems. The platform-independent designs can then be transformed into platform-specific ones to create the operating system-specific products during application engineering [9]. Code generation is also common during application engineering [34].

2.2 UML modeling tools

UML modeling tools offer graphical editors to help architects and developers model requirements, architectures, data structures, dynamic behaviors, and other characteristics of systems. Most tools also support the UML 2 profile mechanism, enabling the creation and use of Domain Specific Languages (DSLs), for example, for variability modeling. Some UML tools can generate software from UML models and UML models from the software. Some modeling tools have a built-in knowledge of UML rules, so they can automatically validate the correctness of UML models. Table 2 presents typical high-level features for the class of UML modeling tools.

UML tools often support distributed software development within and across teams. The traditional approach has been to make model repositories available to the teams through centralized servers. When centralized version management is deployed together with centralized servers, specific locking mechanisms are typically enforced to enable multiple users to simultaneously work with a model and to completely prevent conflicts that otherwise would result from parallel model updates. When more freedom with concurrent model editing is desired, the merging mechanisms of centralized version management systems enable the free concurrent editing of a model, inform developers of possible conflicts when they check their changes into the centralized repository, and merge changes and resolve conflicts automatically or based on developer input.

3. LITERATURE REVIEW

Oldevik et al. [28] propose a set of evaluation criteria for product line modeling tools but the set does not address version management. To our knowledge, no other papers present comprehensive evaluation criteria for product line modeling tools. However, there are a few papers related to the requirements for UML modeling tools [11;23]. This section reviews those papers from the version management point of view to find out how features supporting collaborative work are described.

3.1 General-purpose requirements for the class of UML modeling tools

Funes et al. [11] present a generic set of requirements based on authors' experiences. They provide no references to case studies in particular organizations. They group requirements into the following categories: Features (that are not related to modeling), Modeling support, Customization, Installation and performance, and Tool support. Only three of the requirements relate to the features supporting modeling in collaborative environments (Table 3): (1) Multiple User Support, Access control/sharing, (2) Multiple User Support, Concurrency control, and (3) Versioning. Funes et al. [11] do not explain the requirements in more detail.

Lester & Wilkie [23] propose 15 criteria for UML tool evaluation partially based on the feature lists of existing products. Two of them relate to version management but they are only described as headings and not explained in more detail. The criteria are based on experiences in a software company with only two sites. Therefore, other relevant evaluation criteria might be needed in multi-site and multi-partner organizations.

Feature	*Purpose of the feature is to help*
Modeling & Diagramming	Create, remove, and edit model elements; view the models from different perspectives, and create, remove and edit diagrams.
Hierarchy Management	Create, update, and delete hierarchies (i.e., packages) in which model elements are assigned.
Collaboration and Version management	Multiple concurrent users to manage different versions of assets and to resolve conflicts; integrate the UML tool to version control and/or change management systems as necessary.
Publishing	Compose and publish views of the selected models or model elements; provide data in different formats (XMI, HTML/ODT/PNG/JPG); create reports and documents based on the selected model or model elements.
Tracebility	Create, remove, update, and trace relationships between models or model elements.
Simulation and Validation	Simulate dynamic behaviours of models or interface or integrate the tool to simulation tools; validate UML model correctness and completeness.
Model and Code Synchronization	Generate code based on models; create models based on code (reverse engineering); integrate UML tools to source code systems or Eclipse; integrate UML tools with MDA tools such as oAW, AndroMDA, and BlueAge.
User Management	Manage access and connectivity to the organization's directory services (LDAP, AD).

Table 2. Common features of UML tools.

Funes et al. [11]	Requirements	Multiple User Support 1. Access control/sharing 2. Concurrency control Versioning
Lester et al. [23]	Evaluation criteria	Repository/Version control support Componentization
Oldevik et al. [28]	Evaluation criteria	(None)

Table 3. Version management related requirements for the class of UML tools.

3.2 Summary

The requirements and evaluation criteria for UML modeling tools presented in the papers analyzed in this section do not describe features supporting collaborative modeling in such detail that UML tool evaluations could be completed. In all the papers authors draw upon their own experiences or the feature sets of existing products. Thus, the version management related requirements seem to be based more on the analysis of existing products than on the needs of the users of UML tools. Therefore, this paper will analyze in more depth version management related to product line modeling in multi-site, multi-partner organizations.

4. DESCRIPTION OF THE CASE ORGANIZATION AND THE RESEARCH METHOD

An evaluation framework has been created based on the experiences in the global case organization and the literature review. The case organization is a large multi-site and multi-partner high-tech company using the software product line strategy to successfully operate in highly diverse global markets. The UML modeling tool evaluation project was initiated in 2006 by a department responsible for the development and delivery of global information management solutions and services for R&D units within the case organization. Requirements were gathered during the winter 2007-2008 to understand the features necessary for applying UML modeling tools to model system architectures together with collaborators and partners.

The project was managed according to the internal corporate guidelines for tool evaluation projects. The project team consisted of a project manager, seven architects from major user organizations, and two IT specialists/architects. The first author of this paper was responsible for requirements engineering. Each user organization representative was interviewed by phone during the first phase. Other requirements sources included the industrial best practices reported in journals and in the Internet, modeling tool experts, and IT architects. Requirements were described in writing based on the interviews, reviewed, and prioritized by the project team.

One of the highest priority requirements related to version management was that the UML tools must support sophisticated locking mechanisms. The mechanisms must (1) enable developers to define various parts of a model that they can update independently and (2) prevent conflicts from parallel model updates.

In the first phase, 15 modeling tools from 13 vendors were evaluated. Based on the requirements, three commercial products were selected for in depth evaluations, including detailed vendor liability, financial, and tool architecture evaluations. Details of vendor liability, tool architecture, and financial evaluations are not provided in this paper because the case organization and tool vendors have agreed that the evaluations are confidential. Final evaluations of the three tools were based on feedback from the project team, performance and other tests of the three tool installations in the case organization, reviews with vendors, and the available documentation.

Interestingly, during the evaluation and piloting process it was found out that the modeling tools significantly differed with respect to their version management capabilities. Thus, the case organization became interested in creating a set of more detailed evaluation criteria to enable the detailed analysis of version management capabilities. The criteria presented in Section 5.2 reflect the high-level requirements determined du-

ring the evaluation project and can be used for evaluations of version management capabilities. The detailed questions for each evaluation criteria have been created based on the literature review. The criteria related to the availability of historical traceability information were added to the framework solely based on the experiences in the case organization because historical information has been crucial to ensure proper version management in the organization.

5. TOOL EVALUATION FRAMEWORK

5.1 Assets to be modeled in product line organizations

In a product line organization, the assets the organization creates, maintains, and manages to satisfy market needs constitute systems composed of software and hardware. The hardware and software assets may be managed as products, product lines, and platforms serving several other products or product lines. Other companies, organizations, or individuals may manage and even own the software and hardware components.

Figure 1 depicts a platform shared across two product lines. Both product lines consist of three product variants that are used by markets consisting of individual consumers and/or organizations. Platforms provide common (mandatory) and variable (alternative or optional) features shared across products or product lines. Complex organizational networks can be responsible for owning and sharing the components used within platforms and product lines. In Figure 1, the platform contains three components, which are not managed by the organization responsible for the platform, and product line 1 contains two components, which are not owned or managed by the product line 1 organization.

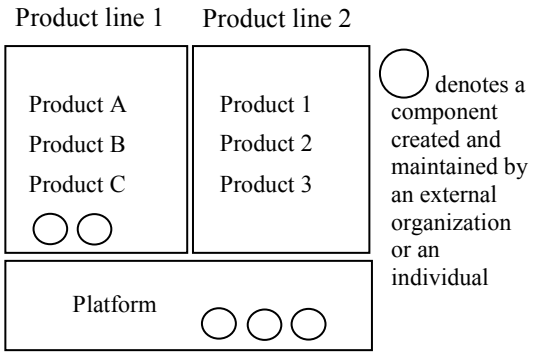

Figure 1. Software and hardware asset design and maintenance responsibility.

The use of models in this network of collaborators and partners would require seamless interactions to share the models. For example, the company responsible for product A may share the understanding of the architecture in the form of models that the partner could further use when planning and implementing the models related to the design of a particular component used in the product.

Different strategies may be implemented to enable modeling of the assets. In this paper we consider a strategy in which one (commercial) modeling tool is used across the product line organization. This strategy minimizes the need for data transfer across tools but may require extensive effort for training and dealing with resistance to organizational change.

5.2 Evaluation characteristics

In this section, the evaluation characteristics are described to define a set of desired version management properties for the class of UML tools (Table 4). The characteristics have been derived from documented version management (e.g., [1]) and product line modeling (Section 5.1) practices and from the requirements, the project team identified in the case organization.

We adopt the following terminology defined by Object Management Group for UML2 [27]:

- A *model* captures a view of a physical system. It is an abstraction of the physical system, with a certain purpose.
- A *package* is used to group elements, and provides a namespace for the grouped elements.
- *Diagrams* are graphical representations of parts of the UML model. UML diagrams contain graphical elements that represent elements in the UML model.
- An *element* is a constituent of a model.
- A *property* is a structural feature.

The justification for each characteristic is indicated by questions to be answered during evaluations. The output domain of permitted answers is also defined for each question. Some questions have *Yes* or *No* as the output domain while others have a range of possible answers.

Two sets of characteristics are defined: one for version management support from the viewpoint of functionality (i.e., which features users can use) and one from the viewpoint of client and server technology to find out whether the technology supporting version management is feasible for large multi-site, multi-partner organizations. Specific questions have been added for each set. Organizations planning to introduce UML modeling tools should carefully consider the questions and add or remove questions according to their specific needs. However, Table 4 serves as a baseline for evaluation. Section 5.2.1 discusses each question in more detail.

Fundamental version management concepts in distributed parallel development of software are check-in/check-out, branching, and merge [1]. System models need to be version controlled in the same manner as software code. For example, when UML is adopted to model the deployment view of software, each model element may have a corresponding element in the software code (see [17] and Section 2.1). As discussed in Section 5.1, different parties may manage the software assets up to the component level. It should thus be possible to manage models up to the element level that corresponds with the component in software. For example, if there is a new version of the component to be branched, the model should reflect this change, so it should be possible to make a branch for the corresponding element in a model. Another example

involves the modeling of a common view of product line architecture (see [4] and Section 2.1). When a team is working on a common view of the architecture through a model and a team member checks out the model, others cannot continue the work until the same person has completed the check-in. During the interviews in the case organization, interviewees reported experiences of model level check-ins and check-outs, which were seen as problematic. The case organization thus determined that it should be possible to check-in and check-out at the element level. Therefore, each version management feature should support operations at the element level.

Version management features	*Evaluation question*	*Evaluation answer*
Check-in/Check-out	Is there support for Model, Package, Diagram and Element level check-in and check-out for multiple users? {Yes/No}	Model, Package, Diagram, and Element level check-in and check-out enable teams to work effectively with the models. Mandatory.
History	Is the Element level history available (who has made what changes)?{Yes/No}	Element level history enables tracing of all the changes. Optional.
Model comparison	Is it possible to compare models at the Element level? {Yes/No}	Element level comparison is a prerequisite for merging. Mandatory.
Merging	Is there support for Model, Package, Diagram, and Element level three-way merge? {Yes/No}	Model, Package, Diagram and Element level three-way merging enables teams to merge models effectively and reliably. Mandatory.
Branching	Is there support for Model, Package, Diagram and Element level branching? {Yes/No}	Model, Package, Diagram and Element level branching enables teams to work effectively with models. Mandatory.
Technologies for version management	*Evaluation question*	*Evaluation answer*
Server-side technology	Are three-tier technologies supported? {Yes/No}	Three-tier technologies enable scalable and reliable solutions.
Client-side technology	Are client installations required? {Yes/No} Are there maintenance needs for the clients? {Yes/No}	Client installations and project-specific needs for tool configurations increase maintenance costs and support needs.

Table 4. Evaluating the version management features and technologies of the class of UML tools.

5.2.1 UML tool features for supporting version management

Check-in/Check-out

Appleton [1] states that most widely used version control tools employ the checkout-edit-checkin model to manage the evolution of version-controlled files in a repository or codebase. Element level check-in and check-out enables completing the necessary tasks effectively in product line modeling. Diagrams, packages, and models could also be useful elements to be checked in and out.

History

Version management requires thorough traceability so users know who has done which changes to the model and can rollback changes if needed. Knowing the history of data helps determine the extents to which the data is trustworthy and up-to-date. Knowing the previous editors also gives points of contact for inquiries. To enable tracebility, log information should be automatically collected and appropriate features should be available to see and analyze the log information. It should be possible to trace back to the element level because users may need to know, for example, who has made changes to a particular component. Diagrams, packages, or models could also be useful elements to trace. However, this feature is optional because users can work without comprehensive tracebility at least as long as their routines and/or tools do not break down. When coordination breakdowns disrupt the routines, it is typically time consuming and expensive to find out and fix the reasons for the breakdowns, if the traceability information is missing [20,21].

Model comparison

Model comparison enables identifying the changes between two models. It can take place in different levels. For example, two models can be compared and their differences can be reported on package, diagram, element, and property levels. Most sophisticated comparison functionalites enable comparisons up to the property level, so users can see the differences between different UML elements' properties. Because efficient merging requires comparisons, this feature is mandatory.

Merging

Merging is the means by which one development line synchronizes its contents with another development line [1]. Merging can be implemented as a 2-way or a 3-way merge. In a 2-way merge, two software artefacts are merged without information about the possible common ancestor. In a 3-way merge, the information about the common ancestor is used. The 3-way merge is more reliable than the 2-way merge because it can detect conflicts better and identify actual changes more precisely. In product line organizations, it should be possible to merge at the element level. For example, users may need to merge models of two branches reflecting changes made to the code. To minimize manual work, diagram merging should also be possible.

Branching

Branching in its most basic form allows development to take place along more than one path for a particular file or directory [1]. Branching can be applied to five different software development situations [1]. Branching of (1) the system's *physical* configuration -

branches are created for files, components, and subsystems, (2) the system's *functional* configuration - branches are created for features, logical changes (bug fixes and enhancements), and other significant units of deliverable functionality (e.g., patches, releases, and products), (3) the system's *operating environment* - branches are created for various aspects of the build and runtime platforms (e.g., compilers, windowing systems, libraries, hardware, and operating systems) and/or for the entire platform, (4) the team's work efforts - *Organizational* branches are created for activities/tasks, subprojects, roles, and groups, and (5) the team's work behaviors - Procedural branches are created to support various policies, processes, and states.

For each category, analogical needs for the branching of product line related models can be identified. (1) Physical branching of models when modeling software for different subsystems as a basis for code generation, (2) functional branching for different products, (3) environmental branching for hardware, software, and related platforms, (4) organizational branching for different projects, and (5) procedural branching to support different product line modeling processes. We see that model branching is analogical to the branching of software and thus it should be possible to branch at least at the package level, as packages provide the mechanism to group model elements. However, optimal support for product line modeling requires branching at the element level. For example, if there is a new version of a component to be branched, it should be possible to make a branch for the corresponding element in the associated model.

5.2.2 Technologies for supporting version management

Version management can be supported by two- or three-tier technologies. Three-tier technologies are more scalable and reliable than two-tier technologies. If client installations are needed, the magnitude of maintenance and support costs incurred to keep the clients updated needs to be considered. If the clients also need to be configured for each modeling project separately, the maintenance and support costs will increase even more.

5.2.3 Summary

This section described a framework consisting of seven criteria to support the evaluation of version management capabilities of UML modeling tools. The criteria were derived from documented version management practices (e.g., [1]) and characteristics needed in product line modeling (Section 5.1). The framework is composed of two sets of characteristics: one from the functional perspective (i.e., which features users are can use?) and one from the technical perspective (i.e., are technologies supporting version management feasible for large multi-site organizations?).

6. USING THE FRAMEWORK TO EVALUATE TWO COMMERCIAL UML MODELING TOOLS

In this section the commercial UML modeling tools Enterprise Architect (http://www.sparxsystems.com/) and Magicdraw (http://www.magicdraw.com/) are evaluated (Table 5) using the framework described in Section 5.2. Commercial UML modeling tools have been selected for the evaluation because global high-tech organizations typically benefit from purchasing commercial modeling tools [26]. Open-source UML tools are not yet as mature as their commercial counterparts are but they have reached a sufficient maturity level to benefit small and medium sized businesses [26]. We chose the two tools for evaluation because they (1) are available for Macintosh, Linux and Windows operating systems, (2) are not too expensive for a large company to deploy for even thousands of users, (3) support SysML, and (4) provide version management features. These four criteria are adequate to simulate a situation where a large company is looking for a UML modeling tool for organization-wide use by both systems and software architects, designers, and other stakeholders.

The term “project” used in both Enterprise Architect and MagicDraw equals to the term “model” adopted in this paper; one project may contain any number of packages, elements and diagrams.

6.1 Enterprise Architect

Enterpise Architect can be used in conjunction with several version management tools such as Subversion, CVS, ClearCase, Visual Source Safe, Accurev, and Perforce. Each package in a model can be version-managed separately as a XMI file, checked-in, modified and checked-out. In addition, *User Security* feature provides means for individual users or user groups to lock, modify, and unlock package(s), diagram(s) or element(s). It is also possible to use several version-managed packages at the same time via *Get-all-latest* feature.

The comparison feature under *Manage Baselines* enables the comparison of models including version-managed packages. Comparison is possible for two models at a time up to the element level including diagrams. Branching can be realized by making a baseline using the *Manage Baselines* feature.

In Enterpise Architect, all clients communicate directly to the centralized version control system via local version management clients. This approach puts pressure on client maintenance because all users need both Enterprise Architect and the version management client installed and configured. Each version managed project needs to be configured separately. Enterprise Architect leverages three-tier technologies; there are three separate processes running (user interface, version management client, and version management server).

Version Management Features	*Evaluation question*	*Enterprise Architect 7.5*	*MagicDraw 16.0*
Check-in/Check-out	Is there support for Model, Package, Diagram and Element level check-in and check-out for multiple users? {Yes/No}	The Model and Package level check-in and check-out and the Element and Diagram level locking and unlocking are supported.	The Model and Package level check-in and check-out and the Element and Diagram level locking and unlocking are supported.
History	Is the Element level history available (who has made what changes)?{Yes/No}	No	No.
Model comparison	Is it possible to compare models at the Element level? {Yes/No}	Yes. Two models can be compared at the Element level including diagrams.	Yes. Two models can be compared at the Element level including diagrams.
Merging	Is there support for the Model, Package, Diagram and Element level three-way merge? {Yes/No}	No. Two-way merge is supported.	Yes. Three-way merge is supported.
Branching	Is there support for Model, Package, Diagram and Element level branching? {Yes/No}	No. Only Model and Package level branching is supported.	No. Only Model and Package level branching is supported.
Version Management Technologies	*Evaluation question*	*Enterprise Architect*	*MagicDraw*
Server-side technology	Are three-tier technologies supported? {Yes/No}	Yes	Yes
Client-side technology	Are client installations required? {Yes/No} Are there maintenance needs for the clients? {Yes/No}	Yes. Yes. Version management tool installation and configuration are needed.	Yes. No. Extra maintenance is needed for Teamwork servers.

Table 5. Comparing the version management features and technologies of MagicDraw and Enterprise Architect.

6.2 MagicDraw

The Teamwork server of MagicDraw allows the assignment of as many developers as necessary to work simultaneously on the same model on multiple workstations. The resulting model is saved and version-managed either on the Teamwork server or in a version management tool connected to the Teamwork server. Currently, MagicDraw can be used with two version management tools: Clearcase and Subversion. Models can be decomposed into sub-models at the package level, enabling model partitioning. Each package can be version-managed separately and checked-in, modified and checked-out. In addition, it is possible to lock, modify, and unlock package(s), diagram(s), and element(s).

Branching is realized at the model level. However, as models can consist of other models (modules), branching can also be considered to work at the package level. Model comparison (*Analyze/Compare projects*) can be used for three-way comparison and merge up-to the element level including diagrams.

No project-specific client configurations are needed for MagicDraw clients. Version management client installations are not needed either. This reduces the need for client maintenance and support. However, the Teamwork servers require extensive maintenance and support. MagicDraw leverages three-tier technologies; there are three separate processes running (the user interface of a MagicDraw client, Teamwork server, and the version management repository server).

7. EVALUATION OF THE FRAMEWORK

Both MagicDraw and Enterprise Architect support the package level check-in and check-out and locking and unlocking up to the element/diagram level. However, in both tools all the changes are saved at the model level. From a technical point of view, check-in/check-out thus requires lots of network traffic, increasing the time needed for check-in/check-out. MagicDraw enables three-way merging while Enterprise Architect provides only two-way merging.

Both MagicDraw and Enterprise Architect enable package level branching. Branching especially in product line organizations should be further studied because package level branching is not seen optimal for product line purposes. For example, if there is a new version of the component to be branched, it should be possible to make a branch for the corresponding element in the associated model. Both Enterprise Architect and MagicDraw lack element level histories. Availability of element level histories would help trace, who has made which changes to a particular element at what time. However, both tools help trace changes by enabling the comparison of models.

From a technical point of view, Enterprise Architect can be used in conjunction with many version management systems, thus being potentially more cost effective. After all, many companies already have comprehensive version management systems. Enterprise Architect requires more maintenance and configuration on the client side (i.e., version management clients need to be installed and configured) whereas MagicDraw requires substantial Teamwork server maintenance. The differences in technology may cause risks in availability and performance and increase maintenance needs. It is thus crucial for organizations to test the real perfor-

mance of the tools by experimenting with a variety of different setups of servers and clients.

Even if the products were quite similar in terms of features, the differences in technologies may increase maintenance needs and pose availability and performance related risks. The use of the framework thus provides essential information to support decision making during the evaluation projects.

Both products can be used in product line modeling because they provide the required basic features. For companies looking for more sophisticated version management features, MagicDraw is the best choice.

7.1 Lessons learnt

The evaluation project in the case organization draws attention to issues, which are general for all organizations considering the adoption of UML modeling tools.

During the project, it was noticed that the usability of the tools' version management features should be further studied because during the piloting phase users need to be spesifically instructed about version management capabilities. Organizations also need to consider the total cost of ownership separately because the possibility to use the already existing version management systems may reduce costs. Organizations planning to introduce UML modeling tools should always consider the framework and, additionally, evaluate usability, efficiency, and the total cost of ownership.

The fact that the two products have similar features also calls for the development of new more innovative solutions. For example, new advances in version management such as Distributed Version Control Systems (DVCS) [29] should be considered.

8. CONCLUSIONS AND FUTURE RESEARCH

The main deliverable of this paper is a framework consisting of a set of criteria for evaluating the version management features of UML modeling tools for multi-site, multi-partner software product line organizations. To illustrate and validate the framework, we applied it to evaluate two UML modeling tools. This study may serve as a baseline to find and implement new product development ideas for improving the UML modeling tools through the design science research [14]. For example, improving the usability of the tools and the capabilities of the users is expected to increase the benefits gained from modeling [3;10].

The results of this study serve as a basis to evaluate features of the UML modeling tools available in the software markets and the relationships between the features and successful deployments. Based on the experiences in the case organization, the deployment projects are more likely to fail if the modeling tools and services do not meet the requirements set in the framework. It is thus crucial to conduct further empirical research to understand better, which tool features will contribute most to the beneficial deployment of the tools.

This paper has focused on evaluating version management features of UML tools that follow the traditional centralized client-server model. However, new tools such as Git have appeared and the dominant ones such as Subversion have been further developed to leverage the Distributed Version Control Systems model challenging the centralized model. These tools operate in a peer-to-peer manner, enabling radical changes in systems development practices. Each developer using such a tool has a copy of the project's entire history and metadata. Developers can share changes in any way that suits their needs, not necessarily through a central server [29]. Although these tools and the enabled practices are not yet robust enough to be used organization-wide by global multi-site, multi-partner corporations, the tools are maturing quickly. Future research is thus needed to assess the applicability of the proposed framework for evaluating DVCS-based UML modeling tools and to revise the framework as necessary.

9. ACKNOWLEDGMENTS

The comments of Andrius Armonas, Erran Carmel, Mitchell Cochran and Rick Kazman greatly improved this paper.

10. REFERENCES

1. Appleton, B., Berczuk, S., Cabrera, R. and Orenstei, R. (1998). Streamed Lines: Branching Patterns for Parallel Software Development. In PLoP '98 conference. Available at: http://www.cmcrossroads.com/bradapp/acme/branching/

2. Arciniegas, J.L., Dueñas, J.C., Ruiz, J.L., Ceron, R., Bermejo, J. (2006). Architecture Reasoning for Supporting Product Line Evolution: An Example on Security. In T. Käkölä & J.C. Dueñas (Eds.), Software Product Lines: Research Issues in Engineering and Management. Springer, 327-372.

3. Arisholm, E., Briand, L.C., Hove, S.E. and Labiche, Y. (2006). The Impact of UML Documentation on Software Maintenance: an Experimental Evaluation, IEEE Transactions on Software Engineering, 32(6), 365 – 381.

4. Bayer, J., Gerard, S., Haugen, Ø., Mansell, J. X, Møller-Pedersen, B., Oldevik, J., Tessier P., Thibault, J-P and Widen, T. (2006). Consolidated Product Line Variability Modeling. In T. Käkölä & J.C. Dueñas (Eds.), Software Product Lines: Research Issues in Engineering and Management. Springer, 195-241.

5. Bertolino, A.., Fantechi, A., Gnesi, S. and Lami, G. (2006). Product Line Use Cases: Scenario-Based Specification and Testing of Requirements. In T. Käkölä & J.C. Dueñas (Eds.), Software Product Lines: Research Issues in Engineering and Management. Springer, 425-444.

6. Brown, S. L. and Eisenhardt, K.M. (1995). Product Development: Past Research, Present Findings, and Future Directions. Academy of Management Review 20(2), 343-378.

7. Carmel, E. and Agarwal, R. (2001). Tactical Approaches for Alleviating Distance in Global Software Development. IEEE Software, 18(2), 22-29.

8. Carmel, E. and Agarwal, R. (2002). The Maturation of Offshore Sourcing of Information Technology Work. MIS Quarterly Executive, 1(2), 65-78.

9. Cusumano, M. A., and Selby, R. W. (1998). Microsoft® Secrets. Free Press, New York, NY.

III

DESIGN, IMPLEMENTATION, AND EVALUATION OF A VIRTUAL MEETING TOOL-BASED INNOVATION FOR UML TECHNOLOGY TRAINING IN GLOBAL ORGANIZATIONS

by

Mervi Koivulahti-Ojala & Timo Käkölä, 2012

Proceedings of the 45rd Hawaii International Conference on Systems Sciences

Design, implementation, and evaluation of a Virtual Meeting Tool-based innovation for UML technology training in global organizations

Mervi Koivulahti-Ojala & Timo Käkölä
University of Jyväskylä
40014 University of Jyväskylä, Finland
{meelheko, timokk}@jyu.fi

Abstract

End-user training is complicated to implement in global corporations whose activities are typically scattered across multiple sites in different countries and leverage information systems in various ways. This is especially true in global software development where the sites may leverage a development tool for totally different purposes. Web-based Virtual Meeting Tools (VMT) enable synchronous communication globally through interactive audio, online chats, video, and the sharing of presentations. They provide potentially a cost effective way to train even complex topics to large numbers of people in global settings. Few industrial experiences from the design and use of VMT-based training innovations have been reported. This paper draws upon a case study in a global corporation to describe the design, implementation, and evaluation of a training innovation, consisting of a set of courses delivered by means of a VMT and conference calls, to support the global deployment of a Unified Modeling Language (UML) modeling tool and to develop UML modeling skills. Evaluation is based on interviews to verify 1) the impacts of the innovation on skills, knowledge and motivation, 2) perceived learner satisfaction with respect to the innovation. The innovation proved successful in improving skills, knowledge, and motivation in the case organization and learners were satisfied with it. Other organizations may benefit from using VMT to train people to use similar complex information systems for supporting global software development.

1. Introduction

End-user training is critical for successful implementation of information systems (IS) (e.g. [5; 7; 26]). End-users need to acquire new knowledge to be able to use new IS applications effectively [3;20]. Deployment of IS is typically accompanied by substantial investments in formal and informal training. The organization and delivery of training is complicated in global corporations and organizational networks. This is especially true in global software development organizations where the development activities are scattered across many sites in different countries, limiting the possibilities for setting up and delivering face-to-face training. The sites may leverage a development tool for totally different purposes, have varying organizational cultures, and employ thousands of end-users with diverse backgrounds.

Web-based Virtual Meeting Tools (VMT) enable synchronous communication globally through interactive audio, online chats, video, and the sharing of presentations. They provide potentially a cost effective way to deliver training in global settings. However, few industrial experiences from the design and use of VMT-based training innovations have been reported. Moreover, prior research indicates that learners are less satisfied with the web-based training when the topic is unfamiliar and complex and more satisfied when using web-based training for learning familiar and non-complex topics like word processing [24]. It is thus unclear whether VMT can support the training of complex software development tasks and tools in a way that learners are satisfied with the training.

This research provides evidence that web-based Virtual Meeting Tools can be designed and implemented to successfully train thousands of end-users so they can complete complex software development tasks with the appropriate tools. The research is expected to be novel as our review of the extant literature did not find any similar earlier research. This paper draws upon a case study in a global corporation to describe the design, implementation, and evaluation of a training innovation, consisting of a set of courses delivered by means of a VMT and conference calls, to support the global deployment of a Unified Modeling Language™ (UML) modeling tool and to develop UML modeling skills.

UML has become an international standard for systems modeling [21]. It is a comprehensive and

complex language, requiring ample, long-term training and learning efforts [8;17]. UML modeling requires the use of versatile UML modeling tools that offer, for example, graphical editors to enable architects, developers, and engineers to model requirements, architectures, data structures, dynamic behaviors, and other characteristics of systems [16]. UML and the supporting modeling tools constitute a critically important technology (hereafter "UML technology") for supporting global software development. This technology, due to its complexity and comprehensiveness, is a challenging domain for training. It is thus an excellent domain of study to determine whether VMT are adequate and scalable to train hundreds or thousands of people to master complex topics in global corporations.

Indeed, the use of UML and UML modeling tools do not automatically lead to productivity improvements. For example, Dzidek et al. [9] found that UML is beneficial when developers must extend non-trivial systems with which they are unfamiliar and that better UML modeling tools and more experienced personnel could yield even larger returns on investment. Productivity improvements from the adoption of the UML technology may not be reached without the cost-effective training of end-users.

The case organization had to find a cost effective way to improve its employees' UML technology related skills, knowledge, and motivation globally. It decided to use Virtual Meeting Tools for UML technology training. However, the extant literature provided the organization with little guidance for designing and implementing such training.

Following the problem-centered approach of Peffers et al. [23], this research was initiated in the case organization to fill the identified gap in knowledge. The research question is as follows:

• Can the UML technology training be organized and delivered through Virtual Meeting Tools in ways that learners are satisfied with the training and the training positively impacts the skills, knowledge and motivation of the learners?

To answer the research question, the four-phased design science research methodology presented by Peffers et al. [23] was deployed. First, *Problem Identification and Motivation* revealed that the UML technology related research did not provide any insights into the design and implementation of VMT innovations for UML technology training. Second, *Objectives for an Innovation* were defined to resolve the problem based on the experiences of the case organization. Third, the key components of the innovation such as content, organization of training, training materials, and trainers' skills and knowledge were *Designed and Developed*. Fourth, learner satisfaction and improvements in skills, knowledge, and motivation were *Evaluated*.

The main contribution of this research is the design, implementation, and evaluation of a VMT-based innovation for UML technology training. Although the design of the innovation has been created based on the experiences in the case organization, we have made every effort to generalize it and to identify potential prerequisites for the innovation, so other organizations can maximally leverage it in UML technology training.

The paper proceeds as follows. Section "Virtual Meeting Tools, Unified Modeling Language and Unified Modeling Language Tools" introduces basic concepts related to VMT, UML modeling, and modeling tools. Section "A Systematic Literature Review" presents research related to UML training and VMT adoption in training. Section "Description of the case organization and the research method" describes the case organization; the objectives for the innovation; the research method; and the innovation (i.e., key features of training such as contents, organization of training, training materials, and trainers skills and knowledge). Section "Preliminary evaluation of the Virtual Meeting Tool-based innovation," details the results of evaluation. Section "Conclusions and Future Research" concludes the paper, addresses the limitations of the conducted research, and provides an outlook to further research.

2. Virtual Meeting Tools and Unified Modeling Language Tools

This section explains the concepts of VMT and UML Tool.

2.1. Virtual Meeting Tools

Virtual Meeting Tools enable real-time interactions through features such as chat tools and audio, video, and user interface screen sharing. They use common browser plug-ins and connect through a local or remote hosting service [10]. Most VMTs are platform independent, allowing users on PCs, Macs, and Linux machines to share identical features [10]. At the appointed times, participants log on to join the sessions.

VMT has been used most extensively in education [10], for example, to arrange remote lectures. But other types of organizations are increasingly using VMT for collaboration and training purposes. For example, individuals can use VMT to collaborate in geographically distributed projects. Learning has become more flexible as VMT has provided more

opportunities for learning at any place. There is often a sense of community even if the collaborators are thousand miles away from each other. Without the time and expense of travel, experts can attend classes from any location and respond to the questions of other participants in real time.

Training through VMT has limitations. Most importantly, the trainers have reduced control over the virtual class-rooms compared to on-site training. As a result, the trainers have to be highly experienced in using VMT to interact effectively with their audiences while missing many visual and other cues.

2.2. Unified Modeling Language tools

Some UML tools can generate software from UML models and UML models from the software. Some also have a built-in knowledge of UML rules, so they can automatically validate the correctness of UML models. Table 1 presents typical high-level features for the class of UML modeling tools.

3. A Systematic Literature Review

Literature was reviewed to verify to which extent existing studies cover VMT usage for UML and UML tool training in industrial settings. To improve the rigor of the study, a systematic literature review was conducted following the principles of Kitchenham et al. [15]. VMT related literature is fragmented and keywords such as e-Learning, online learning, web-based learning, computer-based training, Internet-based training, and web-based training are used. We thus decided to use a broader term "training". UML tool related literature is also fragmented (e.g., using keywords such as "UML tool" or "CASE tool"), so we decided to use a broader term "UML". The following criteria and process were used:

1. The first criterion was to find UML training related articles by searching words "UML" and "training" in title, abstract, and keywords. Decision was based on the title and the abstract of the article.

2. The second criterion was to categorize research according to industrial experiences, that is, whether the research reported industrial experiences or not. The content was visited when it was impossible to determine based on the abstract and the title whether the article reported industrial experiences.

3. The third criterion was to categorize research according to e-Learning, that is, whether the research reported experiences related to e-Learning or not. The content was visited when it was impossible to determine based on the abstract and the title whether the article reported e-Learning related experiences.

Table 1. Main features of UML modeling tools (adapted from [16]).

Feature	Purpose of the feature is to help
Modeling & Diagramming	Create, remove, and edit model elements and diagrams; view the models from different perspectives.
Hierarchy Management	Create, update, and delete hierarchies in which model elements are assigned.
Collaboration and Version management	Multiple concurrent users to manage different versions of assets and to resolve conflicts; integrate the UML tool to version control and/or change management systems as necessary.
Publishing	Compose and publish views of the selected models or model elements; provide data in different formats (e.g, JPG); create reports and documents based on the selected model (elements).
Tracebility	Create, remove, update, and trace relationships between models or model elements.
Simulation and Validation	Simulate dynamic behaviors of models or interface or integrate the tool to simulation tools; validate UML model correctness and completeness.
Model and Code Synchronization	Generate code based on models; create models based on code (reverse engineering); integrate UML tools to source code systems, Eclipse, or Model-driven architecture tools such as AndroMDA.
User Management	Manage access and connectivity to the organization's directory services (e.g., Active Directory).

In the first phase of search, IEEE Explore, ACM Portal, and Elsevier's Science Direct were searched. These databases covered both IS journals and conferences. The number of found articles and related references are described in Table 2. Seven articles related to UML training were found. Three of them reported UML modeling and/or UML modeling tool training in industrial settings. In addition, Virvou and Tourtoglou [28,29] present two potential systems to support UML learning but no industrial experiences were reported. Both Anda et al. [1] and Andersson et al. [2] did not mention usage of any e-Learning tools for UML modeling related training. Bunse et al. [6] reported industrial experiences from the design, organization, and execution of a training program blending e-Learning and face-to-face training to teach 42 employees UML in an automotive branch of a large German corporation. The program started with an online learning phase in

the form of web-based training, in which the learners worked self-directed with the courseware to enhance their knowledge and skills in applying the UML. The phase was a prerequisite for the face-to-face trainings of the second phase. After the face-to-face training, a several weeks long coaching phase concluded the training program. The coach consulted the learners about applying UML in their day-to-day-work. No UML modeling tool related training was included.

Whenever industrial experiences from UML and UML modeling tool training were reported, the importance of organizing UML and UML modeling tool training in a cost effective way was clear ([1], [2]). Anda et al. [1] investigated the adoption of UML modeling principles and tools in a project where a global company applied a UML-based development method in a large, international project with 230 system developers, testers and managers. Adoption was supported by face-to-face training and mentoring. Maximum benefits from UML-based development were not achieved because training (1) was not adapted to the needs of the project and (2) was considered too expensive to provide to project members who were not directly involved with UML-based development. Andersson et al. [2] researched the adoption of UML/SysML modeling principles and tools in an aerospace systems engineering project at Saab Aerosystems. Introducing UML/SysML with a methodology and a supporting toolset in the organization required a clear strategy including just-in-time, face-to-face training and mentor support.

Table 2.Results of literature review.

Search criteria	Number of articles found	Refe-rences
UML training	Seven	[1,2,6,22, 27,28,29]
UML and/or UML tool training in industrial settings	Three	[1,2,6]
Experiences of e-Learning adoption for UML training in industrial settings	One	[6]

Relevant articles may have been published but not found in this literature review. Nevertheless, we conclude that the application of e-Learning in general and VMT in particular for UML modeling and/or UML modeling tool training has not received much attention in the literature.

4. Description of the case organization and the research methodology

4.1. Case Organization

A VMT-supported innovation for UML technology training was implemented in the global high-technology corporation. To support product development, a new UML modeling tool was being rolled out globally. A web-based VMT was provided by the IT department. It was not used for voice sharing but instead employees could use phone lines or a VoIP application to perform conference calls while using the VMT.

4.1.1. Introduction of the UML modeling tool in the case organization

Most of the intended UML tool end-users were from the R&D organization. Other organizations such as partners using the same IT infrastructure were involved as well. Deployment was supported by a team consisting of personnel from the global IT department, the department responsible for process and information systems development and support for R&D, and the subcontractors working for these departments. The system was intended to gradually replace some existing systems and the number of end-users was thus growing.

4.1.2. The need for the Virtual Meeting Tool-based training innovation

The need for a new way of training was noticed based on two surveys conducted in 2009. The middle management responsible for the tool rollout and support decided to conduct the surveys to evaluate how satisfied the end-users were with the tool and the quality of service. The results of the two user satisfaction surveys are presented in detail by [12]. The team analyzed the results of the surveys and concluded that instructions, user guides, and training practices had to be improved. It initiated several improvement activities accordingly. The challenge was that end-users were working in distributed sites while at the same time there was pressure to extend the use of VMT to cut down travelling costs.

The team had previously used VMT in information sharing. Face-to-face trainings were organized in co-operation with the UML tool vendor which provided globally UML technology training and consultancy as well as technical support for their products. However, the vendor had no experience of VMT-based training. The team decided to design and pilot an innovation for UML technology training to improve the overall effectiveness of this training. Since then, the content, material, and organization of

trainings have been iteratively improved based on free-form feedback from end-users. The middle management initiated more formal evaluation during 2010 in the form of a survey but the response rate was unsatisfactory. Interviews were then determined to be the best method to verify that the innovation was viable in 2011 after two years of deployment.

4.2. Research Methodology

4.2.1. Design Science

The first author of this paper was a member of the team responsible for UML tool deployment. Design science research was deemed as the most effective methodology for designing the innovation for UML technology training. Design science is a discipline of information systems research which has recently got ample attention among information systems researchers. Design science research is relevant to practitioners as it aims at solving practical and theoretical problems by creating and evaluating IT artifacts intended to solve identified organizational problems [19;11;23]. The artifacts are the final results of the design process. March and Smith [19] define artifacts as constructs, models, methods and instantiations. There are several extensions to their list of artifacts. Rossi and Sein [25] include the following artifacts: conceptual designs (e.g., definition of a relational model), methods (e.g., design patterns), models and systems (e.g., prototypes and commercial applications), and better theories (e.g., relational algebra). Järvinen [13] includes informational and human resources as potential artifacts, too.

The designed innovation for UML technology training is partly an IT artifact but it also includes human (e.g., trainers, end-users, and their skills, motivations, and stocks of knowledge) and informational resources (e.g., contents of training materials). The case organization experimented with many different ways of supporting UML tool end-users. No other combination of IT artifacts and informational and human resources was found cost effective by the management or appealing by end-users. This paper focuses on the innovation that reflects the only effective combination of the IT artifact (i.e., the VMT tool) and informational and human resources. The innovation is an artifact resulting from the systematic application of the design science methodology.

4.2.2. Design of the Virtual Meeting Tool-based Innovation

The design of the VMT-based training innovation was a result of two years of development work between 2009 and 2011. An initial set of training sessions was created and executed in 2009 in co-operation with the UML tool vendor. The latest set of sessions is introduced in Table 3. A set of sessions was organized typically once every two months. Each set of sessions was delivered during two weeks, so end-users were able to learn the basics within reasonable time. Each session was designed to last between one and two hours, including the time reserved for questions and answers. After each session, feedback was asked via e-mail from participants.

Trainers were not experienced in applying VMT tools for UML technology training when the training was started. They were specialists in both the UML technology and traditional face-to-face training. The UML tool vendor had to make a substantial effort to install and learn to use the VMT tool and the conference call system the case organization had chosen. The vendor then organized training sessions in its physical premises and delivered them via VMT, decreasing the traveling costs of trainers.

Table 3.Names and descriptions of the sessions.

Name of the session	Description of the session
Introduction to UML	Main diagrams of UML, history and evolution of UML language
Introduction to tool	Main features of tool, how to get started with hands-on example, support resources such as Intranet, guides, and IT support
Class diagrams	Class diagram in UML and demonstrations showing how to create class diagrams using the tool
Sequence diagrams	Sequence diagram in UML and demonstrations showing how to create sequence diagrams using the tool
Composite structure diagrams	Composite structure diagram in UML and demonstrations showing how to create such diagrams using the tool
State machine diagrams	State machine diagram in UML and demonstrations showing how to create state machine diagrams using the tool
Use Case diagrams	Use case diagram in UML and demonstrations showing how to create use case diagrams using the tool
Introduction to collaboration	Features to support the collaborative maintenance of UML models using tool (presentation and demonstration)
How to publish models	Features to support the sharing of models in different formats or through Intranet (presentation and demonstration)

Each set of on-line training sessions was advertised through email and Intranet pages. Employees registered in the sessions got personal invitations to the calendar system used in the case organization. However, it was possible to join the sessions without registration because the conference phone number and the link shared during each VMT session remained the same. This flexibility was well received by employees but the VMT technology did not provide the team with possibilities to keep track of the employees joining the sessions. The number of trained employees is thus an estimation based on the invitations sent. 107 employees were invited for the training sessions in 2009 and 150 in 2010. Employees could join the sessions in their offices, in meeting rooms they had reserved, or in other premises, for example, when traveling. They used their mobile phones, conference phones, or PC software (such as VOIP) for conference calls. Both muted and non-muted lines were reserved for calling purposes. When calling to muted lines, they could not make any comments or questions verbally. However, it was possible to send questions or comments through chat to the trainers, who checked the questions and comments and answered them as necessary during the sessions.

Training materials were originally developed for the purposes of face-to-face training. Each concept (e.g., a UML diagram or feature) was introduced first and then the use of the UML modeling tool was demonstrated in the same context. Later on the materials were further developed to better meet the training needs when there is no face-to-face contact. For example, questions were added that trainers could ask to activate learners remotely. Questions charted the ways of using the UML technology (e.g., "Do you use Class Diagram (Yes/No)? Do you find Sequence Diagram useful in your work?") and tested the learners (e.g., "Which one of the following statements is correct?"). It was also possible for the trainers to share information during the sessions about the test results and the opinions of learners. Training material was available in Intranet for end-users to study before, during, or after the training. All the materials followed the same agreed upon way of presentation (e.g., all menu options were presented in *italics*).

4.2.3. Methodology for validating the innovation

The qualitative data was collected through seven interviews after two sets of sessions were organized during June 2011 and September 2011. To keep the interviews informal, semi-structured questions were used. The interviews were conducted over the phone. They were transcribed to a standard format following the semi-structured questions and related themes and sent to the interviewees for review. The transcripts were cross-checked by the research team to capture misunderstandings and potentially missing information. Surveys could not be used for this research despite the substantial number of learners because the response rates for surveys are very low in the case organization. Formalized ways of testing improvements in skills and stocks of knowledge before and after the training sessions (pre-testing and post-testing) were impossible to deploy as the end-users were located all over the world and there were no resources available to collect all the necessary data from them.

To improve the rigor of interviews, the following studies were applied when planning the questions:

- Koivulahti-Ojala and Käkölä [16] for categorizing the ways of using UML modeling tools,
- Kang and Santhanam [14] and Kraiger et al. [18] for identifying potential areas for improvements in skills, stock of knowledge, and motivation,
- Azadeh and Songhori [4] for identifying potential areas of learner satisfaction.

Koivulahti-Ojala and Käkölä [16] proposed that the ways of using UML models for communication can be categorized as follows: human to human, human to machine, machine to human, and machine to machine. Kang and Santhanam [14] identified three knowledge domains that training programs should cover: Application knowledge covering commands and tools embedded in IS applications; business context knowledge covering the use of IS applications to effectively perform business tasks; and collaborative task knowledge covering the task interdependencies between various actors and how the IS application coordinates and mediates these interdependencies (Table 4). End-users of UML modeling tools need to master all the knowledge domains. The business context needs to be mastered because UML modeling tools are general purpose tools applicable to several business processes. Collaborative task knowledge is vital too because UML modeling tools (possibly integrated with other tools) mediate collaborative activities in distributed software development. Table 4 illustrates UML modeling tool related knowledge needs with examples. This study focuses on application and collaborative training as the training sessions supported them. Although Kang and Santhman [14] did not consider motivational aspects in their study, training can positively affect individuals' motivations [18]. Interviews thus charted also motivation issues from application and collaborative task perspective.

Azadeh et al. [4] proposed seven factors that should be taken into account when evaluating end-

Table 4.The model for training users of UML tools (adapted from [14]).

Domain of knowledge	Definition	Example in UML tool context
Application Knowledge 1) Command based 2) Tool-procedural 3) Tool-conceptual	1) Commands/keystrokes needed to execute an operation 2) Knowledge required to combine multiple commands and complete a generic task 3) Knowledge to understand the bigger picture of what to do with a tool	1) Commands/keystrokes in order to create a UML element 2) Combine multiple commands to complete a UML diagram 3) Which types of diagrams should be used together and when, and how the tool facilitates this?
Business context knowledge 1) Business-procedural 2) Business-motivational	1) How to apply the above levels of knowledge to execute a specific business task? 2) What the tool can do for my job? 3) What is the role of the tool in the organization?	1) Which diagrams to apply and when to support a particular business process (e.g., requirements management)? 2) Which business processes of the organization are supported by the UML tool and why?
Collaborative task knowledge 1) Task interdependencies 2) Collaborative problem solving approach	1) Interdependencies between tasks and their effects upon using a UML tool 2) Collaborative problem solving effort between users	1) How tasks completed through the UML tool affect and are affected by other users of the tool (and/or related tools)? 2) Knowledge sharing between users to solve problems

user training programs from learners' perspective: relevance of the course to the learner's job, satisfaction with course content and presentation, quality of instruction, effectiveness of the trainer, and overall satisfaction with the training. Interviews covered all the factors comprehensively.

5. Preliminary Evaluation of the Virtual Meeting Tool-based Innovation

Interviewees' previous knowledge of UML technology varied a lot. Three interviewees had several years of experience of using UML technology and had used this particular UML tool for more than one year. Two had applied UML technology but had used this particular UML tool little or not at all. Two had very little knowledge of UML technology. All of the interviewees shared models with other employees but only one used built-in collaboration functionalities. They did not use models for communication between humans and machines (e.g., code generation or reverse engineering) on a regular basis but some knew such possibilities exist or had even tried using them. Interviewees represented different continents (Asia and Europe) and programmer and architect roles. Most interviewees joined five or more sessions. Those joining less than five sessions were more experienced and wanted to learn specific topics.

5.1. Skills, knowledge and motivation after training

All interviewees were able to name new UML diagrams (or semantics related to a particular UML diagram) or functionalities they had learned during on-line training, indicating that their tool-procedural and tool-conceptual skills had improved. However, the results varied with respect to learning command level skills. An interviewee with limited previous UML technology knowledge mentioned: "If the application is new, you cannot learn everything at one glance." Learners with limited knowledge may thus be overloaded and unable to follow detailed command level instructions. One interviewee had found a solution to support his learning of command level skills. He had completed notes during training so he could later find the right menus more easily. Another interviewee proposed that training sessions should be recorded so the instructions can be reviewed whenever necessary. We can conclude that learners were able to find their ways to learn command level skills over time with the help of on-line training.

Interviewees were not able to name any concepts or practices (e.g., collaborative maintenance of models) related to collaborative task knowledge after training. Only one of the interviewees used collaborative modeling and it can be expected that interviewees focused on those sessions they

considered most relevant to their immediate needs. Thus it cannot be concluded that on-line training is unsuitable for learning collaborative knowledge. Instead, lack of such knowledge after training indicates that interviewees lacked motivation to learn such knowledge. Some interviewees stated that they had not started to use collaborative modeling and therefore had now skipped the related session but were interested to join such a session later.

Most interviewees agreed that they were more motivated to use the UML tool after training. For one user, the usage of the tool was compulsory and he indicated that training neither increased nor decreased his motivation. Another user had a long experience of UML technology and his expectations for the course were learning business-procedural skills and knowledge rather than application level skills. On the other hand, one experienced user indicated increased motivation due to the possibility to refresh his UML technology knowledge. Our preliminary conclusion is that those using the tool voluntarily and joining sessions to learn application level skills were more motivated after the training. Interviewees did not express increased motivations to solve UML technology related problems with other end-users after the training. They mentioned their own teams, Intranet, and Internet as the sources they would use to solve the problems. Training thus improved or maintained motivation at the application level but not at the collaborative level.

In sum, the innovation for UML technology training improved application related skills and knowledge and increased or maintained the motivation to apply UML technology. However, improvements in command level skills and collaborative task knowledge and motivation were limited.

5.2. Learner Satisfaction

Interviewees were satisfied with content, training material, voice, presentation sharing, and the way learning was organized (Table 5). As the content and training materials had been specifically tailored for on-line training of application and collaborative task knowledge during the previous two years, it is possible that the interviewees did not see any major improvement proposals necessary. The proposals for new content came mainly from the users having most extensive previous knowledge. They indicated needs for training either business context related knowledge or very detailed additional knowledge. But additional details might neither be interesting nor useful for novices. Accordingly, the scope of using the VMT innovation must be extended for training business context knowledge in future.

Table 5.Examples of answers for learner satisfaction.

Domain for learner satisfaction	Example answers
Content	"Content was a compact packet." "I think content was good and some good examples were presented. " "Potentially it could go into more details and more advanced things."
Demonstrations	"Demonstrations were clear and presented smoothly." "Good to have the sessions on UML modeling and the use of the tool after each other." "When showing how to make menu selections, the trainer should pause and show the selections slowly."
Training material	"Material was ok". "I have read those materials I need for the three types of diagrams I use." "Good enough whatever there was during those two sessions I attended." "Material does not support finding information. Searching and linking capabilities would be improvements."
Trainer	"Trainer knew the material and the UML tool." "Trainer was fluent in English and knew what he was doing." "He did know his topic and was clear presenting it." "Content was good but sometimes too difficult to follow due to fast speed."
Voice	"No problems." "Mostly ok."
Means of presentation	"No problems". "Surprisingly good. Only one small break."
Questions for the trainer	"It was good to have a chance to make questions. Trainer answered them promptly."
Questions for other interviewees	"Most learners only listened. As far as I remember, one person in two sessions asked something."
The way the on-line training sessions were organized	"A full day session is difficult to allocate nowadays. This [short session] was good for me... I would probably miss it if it were a longer face-to-face or on-line session. On-line sessions are difficult to follow if they last several hours. Face-to-face trainings need full day allocations and negotiations with the manager." "Length of sessions was ok".

One interviewee proposed that it should be possible to find information in the material more easily. The material was tailored for training purposes and did not support the searching of particular pieces of

information. As learners were not able to fully learn command level skills during training sessions, the material should support the searching of relevant content after training.

The organization of the training sessions got some positive remarks. UML was always introduced first and the use of the tool was focused on after that. This combination of tool-conceptual and tool-procedural training was seen beneficial. In addition, the lengths of the sessions were suitable both from practical and learning perspectives (see the last row of Table 4). No interviewees mentioned other types of diagrams that should also be covered in training sessions but some detailed proposals for other topics were mentioned (e.g., how to move elements in a hierarchical model).

Interviewees were mostly satisfied with demonstrations and the trainer but they agreed that presentation speed was sometimes too fast. This is understandable as the trainer could not see learners' reactions and adapt the speed as necessary. On-line training thus requires paying special attention to presentation pace.

Interviewees were familiar with VMT and conference calling. They were satisfied with voice and presentation sharing but stated that sometimes PC applications used for presentation sharing or conference calling were not working properly. However, interviewees knew from their earlier experiences that such incidents happen from time to time. This may explain why the incidents did not decrease their perceived satisfaction. When the studied sets of sessions were organized, all trainers had previous knowledge of applying VMT and voice sharing for training. In organizations where trainers or learners lack similar VMT skills and knowledge, learner satisfaction may be lower than in this organization.

5.3. Generalizable findings from the evaluation

The evaluated innovation for UML technology training is based on experiences from one organization during the period of two years. However, it is possible to make some general recommendations because both the innovation and the related informational and human resources have been specified. VMT can be applied for complex technology training successfully (in terms of learner satisfaction, sustained motivation to use the technology, and improved tool-conceptual and tool-procedural skills and knowledge) in organizations where end-users are familiar with VMT and there are trainers experienced in conducting customized on-line training using the innovation. Organizations, searching for a viable solution for training large numbers of globally distributed employees to use complex software technologies, should thus carefully analyze both employees' and trainers' abilities to use VMT and conference calls.

Subjective opinions of interviewees do not necessarily correlate with real improvements in skills and knowledge or learner satisfaction. However, other data sources within the organization support the interview results. First, a user satisfaction survey completed in the organization indicated that after the UML technology training sessions were initiated, user satisfaction was increased (see details in [12]). Second, the case organization tried out other ways of supporting end-users' efforts to learn UML technology but they were unsuccessful in terms of popularity amongst the end-users.

It should also be noted that in the case organization, both business context knowledge creation and collaborative task knowledge creation were also supported by other means. Business context knowledge creation was supported by UML technology experts who joined deployment projects where teams or projects took the UML tool into use. Experts suggested suitable diagrams, structured the models, and provided tailored training for team-, project-, and department-specific purposes. Collaborative task knowledge creation was enhanced by finding and training contact persons for each team, project, and department, and encouraging the sharing of experiences in user forums.

6. Conclusions and Future Research

This research described an innovation for UML technology training that results from a few years of iterative development of the case organization, content, material, and trainers' skills and knowledge. It was found that VMT can be applied for training people to use complex technologies successfully (in terms of learner satisfaction and motivation and knowledge to use the technology) in organizations where end-users routinely use VMT and there are trainers experienced in on-line training. Information systems professionals benefit from the proposed innovation for UML training when planning, implementing, and evaluating UML training sessions organized through VMT. Information systems management can take advantage of the results when making decisions about VMT usage in complex technology training.

The single case study methodology may not provide a sound basis for generalization. Future research in other organizations is necessary to probe the applicability of VMT in training people to use

especially nontrivial information systems. The UML technology is considered to be complex and difficult to learn. This study indicates that it is possible to support the learning of complex technologies through VMT by structuring the complex content in an appropriate way from the end-users' perspectives.

7. References

[1] Anda, B., Hansen, K., Gullesen, I. and Thorsen, H. (2006). Experiences from introducing UML-based development in a large safety-critical project. Empirical Software Engineering, 11(4), 555-581.

[2] Andersson, H., Herzog, E., Johansson, G. and Johansson, O. (2010). Experience from introducing unified modeling language/systems modeling language at Saab Aerosystems. Systems Engineering, 13(4), 369-380.

[3] Attewell, P. (1992). Technology diffusion and organizational learning: The case of business computing. Organization Science, 3(1), 1-19.

[4] Azadeh, A. and Songhori, M. (2006). End-user training programs planning model based on Information Technology and Information Systems (IT/IS) impact on individual work. In Proceedings of the IEEE Industrial Technology, 2107 -2112.

[5] Bostrom, R., Olfman, L. and Sein, M. (1990). The importance of learning style in end-user training. MIS Quarterly, 14(1), 101-119.

[6] Bunse, C., Grutzner, I., Peper, C., Steinbach-Nordmann, S. and Vollmers, C. (2006). Coaching professional software developers-an experience report. In Proceedings of the 19th Conference on Software Engineering Education and Training, IEEE, 123-130.

[7] Compeau, D., Olfman, L., Sein, M. and Webster, J. (1995). End-user training and learning. Communications of the ACM, 38(7), 24-26.

[8] Dori, D. (2002). Why significant UML change is unlikely. Communications of the ACM, 45(11), 82-85.

[9] Dzidek, W.J., Arisholm, E. and Briand, L.C. (2008). A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. IEEE Transactions on Software Engineering, 34(3), 407-432.

[10] ELI (2006). 7 things you should know about virtual meetings, Educause Learning Initiatives. http://connect.educause.edu/Library/ELI/7ThingsYou ShouldKnowAbout/39388

[11] Hevner, A., March, S., Park, J. and Ram, S. (2004). Design science in information systems research. MIS Quarterly 28(1), 75-105.

[12] Islam, N.A.K.M., Koivulahti-Ojala, M., and Käkölä, T. (2010). A lightweight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool. In Proceedings of the AMCIS 2010.

[13] Järvinen P. (2004). On research methods. Opinpajan kirja, Tampere.

[14] Kang, D. and Santhanam, R. (2003). A longitudinal field study of training practices in a collaborative application environment. Journal of Management Information Systems 20(3), 257-281.

[15] Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J. and Linkman, S. (2009). Systematic literature reviews in software engineering-A systematic literature review. Information and Software Technology, 51(1), 7-15.

[16] Koivulahti-Ojala, M. and Käkölä, T. (2010). Framework for Evaluating the Version Management Capabilities of a Class of UML Modeling Tools from the Viewpoint of Multi-Site, Multi-Partner Product Line Organizations. In Proceedings of the 43rd Hawaii International Conference on Systems Sciences (HICSS-43). IEEE. 1-10.

[17] Kobryn, C. (2002). Will UML 2.0 be agile or awkward? Communications of the ACM, 45(1), 107-110.

[18] Kraiger, K., Ford, J. K. and Salas, E. (1993). Application of cognitive, skill-based, and affective theories of learning outcomes to new methods of training evaluation. Journal of Applied Psychology, 78(2), 311-328.

[19] March, S. and Smith, G. (1995). Design and natural science research on information technology. Decision Support Systems 15(4), 251-266.

[20] Nelson, R., Whitener, E. and Philcox, H. (1995). The assessment of end-user training needs. Communications of the ACM 38(7), 27-39.

[21] Object Management Group (2009). Unified Modeling Language: Superstructure. Formal Specification, version 2.2, 2009.

[22] Pavlov, V. L. and Yatsenko, A. (2005). Using Pantomime in Teaching OOA & OOD with UML. In Proceedings of the 18th Conference on Software Engineering Education & Training, IEEE, 77 -84.

[23] Peffers, K., Tuunanen, T., Rothenberger, M. and Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of Management Information Systems 24(3), 45-77.

[24] Piccoli, G, Ahmad, R., and Ives, B. (2001). Web-based virtual learning environments: A research framework and a preliminary assessment of effectiveness in basic IT skills training. MIS Quarterly, 25(4), 401-426.

[25] Rossi, M. and Sein, M. (2003). Design Research Workshop: A Proactive Research Approach. Presentation delivered at IRIS 26, August 9 – 12, 2003. http://tiesrv.hkkk.fi/iris26/presentation/workshop_designRes.pdf last accessed January 16, 2004.

[26] Sabherwal, R., Jeyaraj, A. and Chowa, C. (2006). Information system success: individual and organizational determinants. Management Science 52(12), 1849-1864.

[27] Turner, S., Perez-Quinones, M. and Edwards, S. (2005). minimUML: A minimalist approach to UML diagramming for early computer science education. Journal on Educational Resources in Computing (JERIC), 5(4), 1-28.

[28] Virvou, M. and Tourtoglou, K. (2006). An Adaptive Training Environment for UML. In Proceedings of the Sixth International Conference on Advanced Learning Technologies, IEEE, 147 -149.

[29] Virvou, M. and Tourtoglou, K. (2006). Intelligent Help for Managing and Training UML Software Engineering Teams. In Proceedings of the Seventh Joint Conference on Knowledge-Based Software Engineering. IOS Press, 11-20.

IV

A LIGHT-WEIGHT, INDUSTRIALLY-VALIDATED INSTRUMENT TO MEASURE USER SATISFACTION AND SERVICE QUALITY EXPERIENCED BY THE USERS OF A UML MODELING TOOL

by

A.K.M. Najmul Islam, Mervi Koivulahti-Ojala & Timo Käkölä, 2010

Proceedings of the AMCIS 2010

A lightweight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool

A.K.M. Najmul Islam
University of Turku
najmul.islam@utu.fi

Mervi Koivulahti-Ojala
University of Jyväskylä
meelheko@jyu.fi

Timo Käkölä
University of Jyväskylä
timokk@jyu.fi

ABSTRACT

The research community has delivered many comprehensive instruments to measure user satisfaction and service quality. However, they may be tedious to deploy in industrial settings, often leading to low response rates. Industrial organizations are thus looking for simpler and more cost effective ways to measure both user satisfaction and service quality. This paper presents and validates a lightweight 8-item instrument to measure the user satisfaction and the quality of service experienced by the users of a Unified Modeling Language tool. The instrument merges ease of use and service-related items. The analysis of the results of two surveys, conducted in a global high-tech corporation, indicates that the instrument has adequate reliability and validity. It is short, easy to use, and appropriate for both practical and research purposes. Future research is needed to validate the instrument in the context of other organizations and other classes of information systems.

Keywords

Service Quality Measurement, User Satisfaction Measurement, UML tools.

INTRODUCTION

Modern business organizations have typically invested ample resources to improve their business processes and Information Technology (IT) infrastructures over the years. During the current economic downturn, most business organizations have continued to increase their IT investments (Kanaracus, 2008) but only in the areas of IT where most business value can be obtained. Organizations thus need to assess the returns of IT investments.

The extant research in information systems (IS) evaluation considers the user satisfaction and the service quality as the central constructs or surrogate measures of the business value of IT. It has produced comprehensive approaches and multi-dimensional instruments (DeLone and McLean, 2003; Petter *et al.,* 2008; Smithson and Hirschheim, 1998; Symons, 1991). However, the instruments are complex and tedious to use in industrial settings. The surveys collect data using so many time-consuming evaluation dimensions that the response rates may deteriorate (Jarrett, 2005; Urbach *et al*., 2009). For example, the widely adopted instrument End User Computing Satisfaction (EUCS) (Doll and Torkzadeh, 1988) deploys 12 questions to measure user satisfaction. If the management also wants to measure service quality using, for example, the IS ZOT SERVQUAL (Kettinger and Lee, 2005), there are 54 additional questions to be answered.

The situation is worsened by the fact that the IT organizations typically offer large portfolios of applications and evaluate all or most of them regularly. For example, the outsourcing of applications and related services is common and the service qualities and applications of all providers must be surveyed frequently to ensure the fulfillment of service level agreements. Because each user is likely to use a substantial portion of the entire portfolio of applications, the same users need to fill numerous lengthy questionnaires to assess the systems and related services. For example, if each user deploys on average ten applications and the IT organization measures each application and related services biannually using EUCS and IS ZOT SERVQUAL, each user should answer 2*10*(12+54) =1320 questions annually. In practice, most users are unlikely to answer all surveys, decreasing the reliability of the results. Finally, the analysis of vast amounts of multi-dimensional data is so cumbersome especially in large organizations that IT departments may find the task insurmountable.

Organizations would thus benefit from lightweight instruments to evaluate the systems and services. They also need to plan sampling and other mechanisms carefully to devise the overall structure for measurement. To address these concerns, this paper draws upon the experiences obtained in a global high-tech corporation that wanted to measure user satisfaction and service quality systematically and organization-wide. The corporation could not accomplish this objective effectively because it experienced all the challenges discussed above. This paper presents and applies a new lightweight instrument containing 8 questions to evaluate a Unified Modeling Language (UML) tool used in the corporation and the services supporting tool deployment. The instrument has been designed to be generally applicable for evaluating a variety of systems and services.

The paper proceeds as follows. Section "Evaluation of user satisfaction and service quality" reviews the research on the measurement of user satisfaction and service quality**.** Section "UML Modeling tools for UML modeling" introduces the basic concepts related to UML modeling and modeling tools. Section "Case organization" describes the case organization and the UML modeling tool used. Section "Research methodology" presents the research methodology and the proposed instrument. Section "Validation of the proposed instrument" presents the preliminary validation. Section "Conclusions and future research" concludes the paper.

EVALUATION OF USER SATISFACTION AND SERVICE QUALITY

User Satisfaction measurement

User satisfaction has received considerable research attention since the 1980s (Bailey and Pearson, 1983; Baroudi *et al.*, 1986; Benson, 1983; DeLone and McLean, 1992; DeLone and McLean, 2002; Ives *et al.,* 1983). It is an important measure of information systems success, often regarded as the easiest and the most useful way to evaluate the IS. Bailey and Pearson (1983, p. 531) define user satisfaction as the "sum of one's positive and negative reactions to a set of factors." Doll and Torkzadeh (1988, p. 261) describe it as "the affective attitude toward a specific computer application by someone who interacts with the application directly." Eagly and Chaiken (1998, p. 296) regard user satisfaction as a "psychological tendency expressed by evaluating a particular entity with some degree of favor and disfavor". Huang *et al.* (2004) conclude that user satisfaction is the most often used construct to measure the success of information systems.

Bailey and Pearson (1983) developed a 39-item instrument to measure user satisfaction of data processing personnel. Ives *et al.* (1983) developed a 39-item User Information Satisfaction (UIS) instrument and a separate 4-item UIS measure using a sample of 200 production managers. Due to some limitations, these instruments are not used as much as the 12-item EUCS instrument (Doll and Torkzadeh, 1988), comprising content, accuracy, format, ease of use, and timeliness factors. EUCS is very comprehensive and addresses most limitations of the previously developed instruments. After the exploratory study was completed in 1988, confirmatory studies with different samples concluded the instrument was valid (Doll *et al.*, 1994; Doll and Xia, 1997). A test-retest of the reliability of the instrument found the instrument was reliable over time (Torkzadeh and Doll, 1991). Harrison and Rainer (1996) showed that the instrument could be used generically to evaluate computer applications. The instrument has become widely adopted and it has served as the reference model for many user satisfaction measurement instruments. Lewis (1995) developed the 19-item Computer Usability Satisfaction Questionnaires to measure system usefulness, information quality, and interface quality. Other authors have developed user satisfaction models for specific areas (e.g, Bargas-Avila *et al*., 2009; Huang *et al.*, 2004; Muylle *et al.*, 2004; Ong and Lai, 2007; Palvia, 1996; Wang and Liao, 2007).

Service quality measurement

Marketing researchers developed the 22-item SERVQUAL instrument to assess service quality through the following five dimensions (Parasuraman *et al.,* 1988):

(1) Tangibles: Physical facilities, equipment, and appearance of personnel;

(2) Reliability: The ability to perform the promised service dependably and accurately;

(3) Responsiveness: The willingness to help customers and provide prompt service;

(4) Assurance: The knowledge and courtesy of employees and their ability to inspire trust and confidence; and

(5) Empathy: Providing caring and individualized attention to customers.

SERVQUAL has been adopted in a variety of domains such as healthcare, education, banking, financial services and IS (e.g., Jiang *et al.,* 2002; Pitt *et al.,* 1995). Nyeck *et al.* (2002, p. 102) stated the SERVQUAL instrument "remains the most complete attempt to conceptualize and measure service quality." In the IS field the application of the instrument has garnered a great deal of debate recently (for a review of most debated issues, see (Landrum *et al.,* 2009)). The case organization did not find SERVQUAL attractive for two reasons. First, SERVQUAL includes only one training and documentation related

question: "Useful support materials (such as documentation, training, videos, etc.)". Yet, the role of documentation is emphasized in the context of open source tools because nobody may be supporting these tools. Second, when the support is centralized, the users may not be able to meet the support personnel face-to-face in order to evaluate physical facilities, equipment, or personnel-related tangibles. Therefore, SERVQUAL may not be attractive when open source tools are used or the support organization is centralized.

UML MODELING TOOLS FOR UML MODELING

Unified Modeling Language™ has become an international standard for systems modeling (ISO, 2005). UML modeling tools offer graphical editors to enable architects, developers, and engineers to model requirements, architectures, data structures, dynamic behaviors, and other characteristics of systems. UML models can be used to support communication between people, document a system, generate test cases, predict the realized system's quality, and automate code generation. UML tools may generate software from the UML models and UML models from the software (reverse engineering) and may have a built-in knowledge of UML rules to validate the correctness of the models automatically. Table 1 presents high-level features for the UML modeling tools (adapted from Koivulahti-Ojala and Käkölä, 2010).

The use of UML and UML modeling tools do not automatically lead to productivity improvements. Their potential may not be reached, if engineers need to struggle with the problems related to the poor availability or usability of modeling tools or the lack of user support and training. For example, Arisholm *et al.* (2006, p. 365) studied the impact of UML documentation on software maintenance and concluded that "for complex tasks and past a certain learning curve, the availability of UML documentation may result in significant improvements in the functional correctness of changes as well as the quality of their design. However, there does not seem to be any saving of time. For simpler tasks, the time needed to update the UML documentation may be substantial compared with the potential benefits, thus motivating the need for UML tools with better support for software maintenance." Dzidek *et al.* (2008) found that using the UML could be beneficial when a developer must extend a nontrivial system with which he/she is unfamiliar and that better UML tools and more experience would likely yield even a larger return on investment. These results indicate that when the processes and capabilities are improved through, for example, better UML tools, training, and user support, returns on UML-related investments can be substantial. Measuring user satisfaction and service quality is crucial to focus the required improvement actions appropriately.

CASE ORGANIZATION

This research project was conducted in a global high-technology corporation, developing products in multiple sites with multiple partners. To support product development, a new UML modeling tool was being rolled out globally when the research project started. Most of its users were from the R&D organization. It was supported by a virtual team consisting of personnel from the global IT department and the department responsible for process and information systems development and support for R&D as well as subcontractors working for these departments. The middle management responsible for the tool rollout and support decided to conduct two surveys to evaluate how satisfied the users were with the tool and the quality of service. The tool was intended to gradually replace some existing tools. Numerous users thus adopted the tool between the two conducted surveys. The section "Research methodology" describes the process of study design. The name of the UML tool selected for rollout is not disclosed here. The main functionalities of the tool are presented in Table 1.

RESEARCH METHOLODOGY

Study design

Two surveys were conducted. Table 2 provides their sample details. The email invitations were sent to all the people who had registered as users by the date of each survey. One reminder was sent to the same users.

Instrumentation

The instrumentation of the survey was developed in co-operation with the virtual team responsible for tool support and deployment. The team had three main requirements for the instrumentation: 1) it should measure both the service quality and the user satisfaction with respect to the tool; 2) there should be no more than 10 questions, 3) the survey should be applicable to develop the service and the tool further together with the tool vendor. The first requirement limited the possibility to use a standard survey as to our knowledge there is no standard survey to cover both the service quality and the tool related satisfaction. The authors of this paper created a new instrument, which was accepted by the case organization. The list of questions in the instrument is given in Appendix. Identifiers (Q1-Q11) express the questions in short form. Q8, "Overall, how satisfied are you with <UML Modeling Tool> tool and service" was included for use as the criterion for data analysis because it covers both the service quality and the user satisfaction with respect to the tool. A five scale measure was used from '5 =

Very Satisfied' to '1 = Very Dissatisfied' for questions, Q1-Q8. In our data collection, we randomized the questions in the instrument, mostly eliminating the common method bias (Straub *et al.,* 2004).

Feature	*Purpose of the feature is to help*	*Functionalities that the UML modeling tool in the case organization supports:*
Modeling & Diagramming	Create, remove, and edit model elements and diagrams; view the models from different perspectives.	**Yes.** Create, remove and edit of the following UML diagrams: Use Case, Class, Object, Composite Structure, State Machine, Protocol State Machine, Activity, Sequence, Communication, Component, and Deployment Diagrams
Hierarchy Management	Create, update, and delete hierarchies in which model elements are assigned.	**Yes.** Possible to create a package hierarchy.
Collaboration and Version management	Multiple concurrent users to manage different versions of assets and to resolve conflicts; integrate the UML tool to version control and/or change management systems as necessary.	**Yes.** Integration to version control which enables multiple users to manage models concurrently.
Publishing	Compose and publish views of the selected models or model elements; provide data in different formats (e.g, JPG); create reports and documents based on the selected model (elements).	**Yes.** Possibilities such as report generation, publishing in the HTML format, and copying diagrams in different formats. Open Application Programming Interface for accessing models. XML Metadata Interchange and Eclipse Modeling Framework support model interchange.
Tracebility	Create, remove, update, and trace relationships between models or model elements.	**Yes.** Possibility to create relationships between model elements and trace those relationships.
Simulation and Validation	Simulate dynamic behaviors of models or interface or integrate the tool to simulation tools; validate UML model correctness and completeness.	**Limited**. No simulation possibilities for dynamic behaviors. Validation of UML models is possible (Object Constraint Language or Java).
Model and Code Synchronization	Generate code based on models; create models based on code (reverse engineering); integrate UML tools to source code systems, Eclipse, or Model-driven architecture tools such as AndroMDA.	**Yes**. Code generation/reverse engineering: (e.g., Java 5, EJB 2.0). Integration with Integrated Development Environments.
User Management	Manage access and connectivity to the organization's directory services (e.g., Active Directory).	**No.** However, integrated version control system may be connected to directory services.

Table 1. Main features of UML modeling tools (adapted from Koivulahti-Ojala and Käkölä, 2010)

Survey	Number of invitations	Number of responses (N)	Percentage of responses
Survey 1	267	42	15.73%
Survey 2	444	62	13.96%

Table 2. Sample data

Actions taken in the case organization

The virtual team supporting the UML Modeling tool analyzed the results of the surveys. As the validation results were not available during that time, the team made decisions based on the means of all questions and the total mean of all questions. Based on the 1st survey, communication and training practices had to be improved because the means of questions related to instructions, user guides, and training were lower than the mean of all questions.

Based on the 1st survey, information sharing with the users was improved in several ways and training sessions were organized. Information letters were emailed to the users, new guides were created, and the Intranet pages providing information about the tool and related support were improved. Tens of users were trained in on-line and face-to-face training sessions before the second survey was organized. Conference calls and virtual meeting tools were used, respectively, to share voice and presentations in on-line training sessions.

The answers to the feedback question Q11 were analyzed together with the tool vendor. In 1st and 2nd surveys, respectively, 20 and 18 users gave feedback. A requirements management process and tool were used to manage the UML tool related requirements sourced from the answers.

The results of the second survey revealed that the improvements related to information sharing and training had raised user satisfaction and that the availability and speed of the tool would be the next areas to improve. Fortunately, the software upgrades had already been planned to increase the reliability and usability of the version management features and to make the features faster to use. No separate action plan was thus necessary.

VALIDATION OF THE PROPOSED INSTRUMENT

This section presents the univariate and bivariate analyses for the two surveys. The PASW 18.0 software was used for data analysis.

Central tendency computation

All the questions in the study are either nominal or ordinal. The central tendency of nominal/ordinal variables can be best explained by the Median and Mode (Bryman and Cramer, 1999). Besides them, the mean, standard deviation, and range of all the questions are presented in Table 3.

Question	Mean	Median	Mode	Std	Range
Q1	3.79[1], 3.52[2]	4[1], 4[2]	4[1], 4[2]	.782[1], .880[2]	3[1], 4[2]
Q2	4.00[1], 4.03[2]	4[1], 4[2]	5[1], 4[2]	1.036[1], .849[2]	3[1], 3[2]
Q3	3.79[1], 3.57[2]	4[1], 4[2]	4[1], 4[2]	.871[1], .819[2]	3[1], 4[2]
Q4	3.51[1], 3.63[2]	4[1], 4[2]	4[1], 4[2]	.746[1], .821[2]	3[1], 3[2]
Q5	3.93[1], 4.02[2]	4[1], 4[2]	4[1], 4[2]	.877[1], .833[2]	4[1], 4[2]
Q6	3.30[1], 3.92[2]	3[1], 4[2]	4[1], 4[2]	.966[1], 1.01[2]	4[1], 4[2]
Q7	4.12[1], 4.00[2]	4[1], 4[2]	4[1], 4[2]	.803[1], .923[2]	3[1], 4[2]
Q8	3.88[1], 4.00[2]	4[1], 4[2]	4[1], 4[2]	.739[1], .810[2]	3[1], 3[2]

1: Survey 1, 2: Survey 2

Table 3. Central tendency computation

Linear Regression Method

In order to ensure statistical conclusion validity (Straub *et al.,* 2004), we perform regression analysis. The regression analysis assumes Q8 (criterion) is the dependent variable and the others (Q1-Q7) are independent variables. Table 4 provides the results of the regression analysis.

Question	R-Squared	Constant	B
Q1	.142[1], .305[2]	2.533[1], 2.208[2]	.356[1], .508[2]
Q2	.366[1], .128[2]	2.154[1], 2.623[2]	.432[1], .341[2]
Q3	.142[1], .244[2]	2.671[1], 2.253[2]	.320[1], .489[2]
Q4	.268[1], .242[2]	2.053[1], 2.472[2]	.520[1], .440[2]
Q5	.100[1], .305[2]	2.904[1], 2.113[2]	.248[1], .486[2]
Q6	.012[1], .260[2]	4.088[1], 2.623[2]	-0.80[1], .369[2]
Q7	.466[1], .524[2]	1.292[1], 1.462[2]	.628[1], .635[2]

1: Survey 1, 2: Survey 2

Table 4. Regression analysis results

The following rule proposed by Bryman and Cramer (1999) is followed in identifying how well each question fits the data:

- <0.1: poor fit
- 0.11– 0.3: modest fit
- 0.31– 0.5: moderate fit
- > 0.5: strong fit

Table 4 shows there is at least the modest fit for all questions except Q6 in both surveys. The R squared values for Q6 in the 1st and 2nd surveys are, respectively, 0.012 (poor fit) and 0.260 (modest fit). It means that the overall satisfaction is not explained by Q6 in the 1st survey because people were not satisfied with the available training or training had low importance in measuring overall satisfaction. However, the 2nd survey suggests that training impacted the overall satisfaction. People were not satisfied with the training in the first survey and their overall satisfaction level was mainly caused by other areas (Q1-Q5 and Q7). The low satisfaction level of training revealed by the 1st survey is also visible from the mean of Q6 which is 3.30 while in the 2nd survey the mean is 3.92 (Table 3). The difference may be explained by the fact that both on-line and face-to-face training sessions were arranged between the surveys. The strongest fit is observed for Q7.

Item to Criterion correlation

In order to ensure the criteria-related validity (Boudreau *et al.*, 2001), the correlation of each item with the overall criterion is computed. Table 5 shows the correlation coefficients. Some prior studies (e.g., Doll and Torkzadeh, 1988) suggest having a cut-off point as 0.40 for this criteria-related validity check. Table 5 shows most of the correlation results are above the cut-off point. However, the coefficient for Q5 in the first survey is slightly below the cut-off point. On the other hand, the correlation coefficient of Q6 in the first survey is very low (also confirmed by the regression method). The explanation to this was given in the previous subsection.

Question	Correlation Coefficient
Q1	.426[1], .488[2]
Q2	.621[1], .406[2]
Q3	.422[1], .474[2]
Q4	.532[1], .491[2]
Q5	.390[1], .544[2]
Q6	.042[1], 447[2]
Q7	.719[1], 669[2]

1: Survey 1, 2: Survey 2

Table 5. Item to Criterion correlation

Item to total correlation

To ensure higher model reliability, the correlation of each item's score with the total of all items' scores has been computed. A threshold of 0.45 is used for this validity check. Table 6 shows that the correlation values are well above the threshold except the result of Q6 in the 1st survey (see the explanation in 'Linear Regression Method' subsection).

Factor analysis

The factor analysis was performed only for the data from the second survey that had enough responses. The principle component analysis was used as the extraction technique and varimax was used as the method of rotation. Two formative factors (Petter *et al.*, 2007) were revealed with eigenvalues greater than 1.00, explaining about 61% of the total variance: System Use and System & Support Richness. The item loadings are given in Table 7. Some prior studies (Ong and Lai, 2007; Bargas-Avila *et al.*, 2009) suggested using 0.5 as the threshold value for the item loadings. All item loadings are above the threshold, except the Q2 loadings. Q2 represented both factors to some extent, demanding some more validation of the instrument using more data. The Cronbach's alphas for the factors were 0.65 and 0.792 respectively.

Question	Correlation Coefficient
Q1	.458[1], .569[2]
Q2	.549[1], .634[2]
Q3	.683[1], .618[2]
Q4	.600[1], .708[2]
Q5	.544[1], .737[2]
Q6	.372[1], .756[2]
Q7	.734[1], .616[2]

1: Survey 1, 2: Survey 2

Table 6. Item to total correlation

Question/ Item	Factor 1 (System Use)	Factor 2 (System & Support Richness)
Q1	**.699**	
Q2	.421	.437
Q3	**.888**	
Q4		**.559**
Q5		**.848**
Q6		**.808**
Q7		**.749**

Table 7. Rotated Factor Matrix

Test-retest reliability

Based on the central tendency computation and the regression and correlation-based analyses, both surveys provide similar results and relationships, thus confirming the test-retest reliability check. However, there were some exceptions due to a limited number of responses in the first survey and lack of training and communications.

CONCLUSIONS AND FUTURE RESEARCH

The extant literature provides few, if any, methodologies and instruments that could be used effectively to measure user satisfaction with respect to applications and services in industrial contexts where the effective execution of business processes is dependent on the use of tens of application systems. New instruments are thus needed that enable IT organizations on a regular basis (i.e., even several times a year) to measure user satisfaction with respect to all the applications and related services that belong to the portfolios of the IT organizations.

This paper presents a lightweight 8-item instrument, merging ease of use and service-related items, to measure user satisfaction with respect to both an application and related services. Based on the use of the instrument in one organization to assess user satisfaction with respect to one application and the related services, the instrument appears to have adequate reliability and validity. It is easy to use and appropriate for both practical and research purposes. The case organization was able to plan and implement improvements by analyzing the means of all questions. We thus encourage practitioners to adapt and test the instrument in their own application and service contexts and academics to further validate and refine the instrument in different organizations and for a variety of classes of systems.

REFERENCES

1. Arisholm, E., Briand, L.C., Hove, S.E., Labiche, Y. (2006) The impact of UML documentation on software maintenance: An experimental evaluation, *IEEE Transactions on Software Engineering*, 32, 6, 365-381.
2. Bailey, J. E. and Pearson S. W. (1983) Development of a tool for measuring and analyzing computer user satisfaction, *Management Science*, 29, 5, 530-545.

3. Bargas-Avila, J. A., Lötscher, J., Orsini, S. and Opwis, K. (2009) Internet satisfaction questionnaire: Development and validation of a questionnaire to measure user satisfaction with the Internet, *Computers in Human Behavior*, 25, 1241-1250.
4. Baroudi, J. J., Olson, M. H., Ives, B. (1986) An empirical study of the impact of user involvement on system usage and information satisfaction, *Communications of the ACM* 29, 3, 232-238.
5. Benson, D. H. (1983) A Field Study of End-User Computing: Findings and Issues, *MIS Quarterly* 7, 4, 35-45.
6. Boudreau M.-C., Gefen, D., and Straub D. W. (2001) Validation in information systems research: A state-of-the-art assessment, *MIS Quarterly*, 25, 1, 1-16.
7. Bryman, A. and Cramer, D. (1999) Quantitative data analysis with SPSS release 8.0 for Windows: For Social Scientists, Routledge, New York.
8. Jarrett, C. (2005) Survey Response Rates? 2% is not good enough. (http://www.usabilitynews.com/news/article2528.asp)
9. Delone, W. and Mclean, E. (1992) Information systems success: the quest for the dependent variable, *Information Systems Research* 3, 1, 60-95.
10. Delone, W. and McLean, E. (2002) Information systems success revisited, in *Proceedings of the 35th Hawaii International Conference on Systems Sciences*, IEEE Computer Society, Hawaii, USA.
11. Delone, W. and McLean, E. (2003) The DeLone and McLean model of information systems success: A ten year update, *Journal of Management Information Systems* 19, 4, 9-30.
12. Doll, W. J. and Torkzadeh, G. (1988) The measurement of end user computing satisfaction, *MIS Quarterly* 12, 2, 259-274.
13. Doll, W. J. and Xia, W. (1997) A confirmatory factor analysis of end user computing satisfaction instrument: A replication, *Journal of End User Computing* 9, 2, 24-31.
14. Doll, W. J., Xia, W. and Torkzadeh, G. (1994) A confirmatory factor analysis of end user computing satisfaction instrument, *MIS Quarterly* 18, 4, 357-369.
15. Dzidek, W.J., Arisholm, E. and Briand, L.C. (2008) A realistic empirical evaluation of the costs and benefits of UML in software maintenance, *IEEE Transactions on Software Engineering*, 34, 13, 407 – 432.
16. Eagly, E. A. and Chaiken, S. (1998) Attitude structure and function, *Handbook of social psychology* (4th ed., 269-322), Mc Graw-Hill, New York.
17. Harrison, A. W. and Rainer, K. R. (1996) A general measure of user computing satisfaction, *Computers in Human Behavior*, 12, 1, 29–92.
18. Huang, J.-H., Yang, C., Jin, B.-H. and Chiu, H (2004) Measuring satisfaction with business-to-employee systems, *Computers in Human Behavior*, 20, 17-35.
19. ISO/IEC 19501 (2005) Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2. International Organization for Standardization.
20. Ives, B., Olson, M. H. and Baraoudi, J. J. (1983) The measurement of user information satisfaction, *Communications of the ACM,* 26, 10, 785-793.
21. Jiang, J.J., Klein, G. and Carr, C. L. (2002) Measuring information systems service quality: SERVQUAL from the other side, *MIS Quarterly,* 26, 2, 145-166.
22. Kanaracus C. (2008) Gartner: Global IT sending growth stable. InfoWorld.
23. Kettinger, W. J. and Lee, C. C. (2005) Zones of tolerance: Alternative scales for measuring information systems service quality, *MIS Quarterly*, 29, 4, 607-623.
24. Koivulahti-Ojala, M. and Käkölä, T. (2010) Framework for evaluating the version management capabilities of a class of UML modeling tools from the viewpoint of multi-site, multi-partner product line organizations, in *Proceedings of the 43rd Hawaii International Conference on Systems Sciences*, IEEE Computer Society, Hawaii, USA.
25. Landrum, H., Brybutok, V., Zhang, X., Peak, D. (2009). Measuring IS system service quality with SERVQUAL: Users' perceptions of relative importance of the five SERVPERF dimensions, *Informing Science: the International Journal of an Emerging Transdiscipline*, 12.
26. Lewis, J. R. (1995) IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use, *Human-Computer Interaction*, 7, 1, 57–78.

27. Muylle, S., Moenaert, R. and Despontin, M. (2004) The conceptualization and empirical validation of web site user satisfaction, *Information & Management*, 41, 5, 543–560.
28. Nyeck, S., Morales, M., Ladhari, R., and Pons, F. (2002) 10 years of service quality measurement: reviewing the use of the SERVQUAL instrument, *Cuadernos de Difusion*, 7, 13, 101-107.
29. Ong, C.S., Lai, J.Y. (2004) Developing an instrument for measuring user satisfaction with knowledge management systems, in *Proceedings of the 37th Hawaii International Conference on Systems Sciences*, IEEE Computer Society, Hawaii, USA, 1-10.
30. Palvia, P. C. (1996) A model and instrument for measuring small business user satisfaction with information technology, *Information & Management*, 31, 151–163.
31. Parasuraman, A., Zeithaml, V.A., and Berry, L.L. (1988) SERVQUAL: A multiple-item scale for measuring consumer perceptions of service quality, *Journal of Retailing*, 64, 1, 12-37.
32. Pitt, L.F., Watson, R. T. and Kavan, C.B. (1995), Service quality: A measure of information systems effectiveness, *MIS Quarterly,* 19, 2, 173-187.
33. Petter, S., Straub, D. W., and Rai, A. (2007) Specifying formative constructs in information systems research, *MIS Quarterly,* 31, 4, 623-656.
34. Petter, S., Delone, W. and Mclean, E. (2008) Measuring information systems success: Models, dimensions, measures, and relationships, *European Journal of Information Systems,* 17, 236-263.
35. Smithson, S. and Hirschheim, R. (1998) Analyzing information systems evaluation: another look at the old problem, *European Journal of Information Systems*, 7, 158-174.
36. Straub, D. W., Boudreau, M.-C., and Gefen, D. (2004) Validation guidelines for IS positivist research, *Communications of the Association for Information Systems,* 13, 380-427.
37. Symons, V. J. (1991) A review of information systems evaluation: content, context and process, *European Journal of Information Systems*, 1, 3, 205-212.
38. Torkzadeh, G. and Doll, W. J. (1991) Test-retest reliability of the end-user satisfaction instrument, *Decision Sciences*, 22, 1, 26-37.
39. Urbach, N., Smolnik, S. and Riemp, G. (2009) Development and validation of a model for assessing the success of employee portals, in *Proceedings of the 17th European Conference on Information Systems*, Verona, Italy.
40. Wang, Y. and Liao, Y. (2007) The conceptualization and measurement of m-commerce user satisfaction, *Computers in Human Behavior*, 23, 1, 381–398.

APPENDIX

Q1. How satisfied are you with the speed of <UML Modeling Tool>?

Q2. How satisfied are you with the availability of <UML Modeling Tool>?

Q3. How satisfied are you with the ease of use of <UML Modeling Tool>?

Q4. How satisfied are you with the instructions and user guides available for <UML Modeling Tool>?

Q5. When needed, I get support fast and in a professional way

Q6. How satisfied are you with training available for <UML Modeling Tool>?

Q7. How well does <UML Modeling Tool> tool meet your modeling needs?

Q8. Overall, how satisfied are you with <UML Modeling Tool> tool and service?

Q9. How often do you use <UML Modeling Tool> (Weekly, Daily, Monthly, Less than Monthly)?

Q10. Your area is (EMEA, APAC, Americas)

Q11. Please give feedback (E.g.Improvements, development ideas)

V

TRAINING PEOPLE TO MASTER COMPLEX TECHNOLOGIES THROUGH E-LEARNING: CASE OF UML TECHNOLOGY TRAINING IN A GLOBAL ORGANIZATION

by

Mervi Koivulahti-Ojala & Timo Käkölä, 2014

Proceedings of the AMCIS 2014

Training people to master complex technologies through e-Learning: Case of UML technology training in a global organization

Completed Research Paper

Mervi Koivulahti-Ojala
University of Jyväskylä
meelheko@jyu.fi

Timo Käkölä
University of Jyväskylä
timokk@jyu.fi

Abstract

E-Learning tools facilitate asynchronous (e.g., wiki) and synchronous (e.g., video-conferencing) learning. They may provide a cost effective way to train large numbers of people in global settings to leverage complex technologies such as Unified Modeling Language (UML) tools. Few industrial experiences from the use of e-Learning tools to train complex technologies have been reported. It is unclear whether e-Learning tools can be deployed to learn three types of knowledge: application, business context, and collaborative task knowledge. This paper draws upon a case study in a global corporation to evaluate the applicability of several classes of e-Learning tools for supporting the learning of application, business context, and collaborative task knowledge required to deploy a UML modeling tool globally. Intranet, e-mail, and a Virtual Meeting Tool (VMT) proved most beneficial for learning the three types of knowledge. Other organizations may benefit from using these classes of e-Learning tools to train people to use similar complex technologies in global organizations.

Keywords

E-Learning, training, global software development, UML, UML tool

Introduction

User training is critical for successful deployment of information systems (IS) (e.g., Attewell, 1992; Bostrom et al., 1990; Compeau et al., 1995; Nelson et al., 1995). Collaborative IS applications improve organizational coordination and collaboration among users to complete business processes and have extensive built-in control enforcing standard and shared work practices (Kang and Santhanam, 2003). They are increasing in complexity and need to be deeply engrained in organizational coordination, collaboration, and business practices. Users need to learn application, business context, and collaborative task knowledge to use collaborative IS applications effectively (Kang and Santhanam, 2003). Meeting these requirements is nontrivial and potentially expensive. Organizations are looking for new ways to accelerate learning and to cut IS training costs.

Using E-Learning tools is a way to meet the requirements. Organizations often routinely deploy tools to support asynchronous (e.g., wiki, discussion forums) and synchronous (e.g., video-conferencing, chat) meeting and collaboration practices. If they can use the same tools for training and learning complex technologies, little, if any, additional deployment investments for e-Learning tools are typically needed. However, few industrial experiences from the use of these tools to train complex technologies have been reported. Some prior research results indicate that e-Learning may be ineffective when the topics are complex. According to Piccoli et al. (2001) learners were less satisfied with e-Learning, when learning unfamiliar and complex topics, and more satisfied when learning familiar and non-complex topics like word processing. Hrastinski (2008) found that e-Learning is suitable for discussing and reflecting even complex topics but may need to be complemented with face-to-face meetings. It is unclear whether e-Learning can be successfully used as the only means to support the training of complex technologies in a

way that learners are satisfied with the training and learn application, business context, and collaborative task knowledge.

This paper answers the following research question: which classes of e-Learning tools are the most applicable ones for organizing and delivering technology training scalably so that large numbers of learners learn application, business context, and collaborative task knowledge in order to master collaborative IS applications and other similar complex technologies? This research draws upon a case study in a global high-tech corporation, deploying complex technologies corporate-wide, to provide insights into effective industrial practices for applying e-Learning in the training of complex technologies. We expect the practices to be generalizable to other organizations leveraging complex technologies.

Literature review

This section introduces the concepts of e-Learning and UML technology and analyzes the limitations of the current UML technology training research in industrial settings.

UML technology

Unified Modeling Language™ (UML) is an international standard for object-oriented systems modeling (Larman, 2005, p. 10). It offers 14 types of diagrams and numerous symbols (ISO/IEC, 2012). UML modeling tools offer graphical editors for modeling systems. Tools generate software from UML models and create UML models from the software, have a built-in knowledge of UML rules, and support collaborative software development. Therefore, they are collaborative IS applications. UML and the modeling tools constitute an important technology (hereafter "UML technology") for supporting global R&D. The extant literature provides few lessons for using e-Learning tools in UML technology training in industrial settings (Koivulahti-Ojala and Käkölä, 2012).

We consider UML technology a complex technology due to the following reasons. First, there is a high number of diagrams and symbols that learners need to get familiar with. Second, modeling requires both the understanding of UML and the ability to use the modeling tools. Third, using UML requires long-term training and learning investments (Dori, 2002; Kobryn, 2002).

E-Learning

E-Learning comprises all forms of information and communication technology (ICT) mediated learning (Tavangarian et al., 2004). It can be used alone or to complement other training modes. E-Learning tools serve as a media to implement the learning process (Tavangarian et al, 2004). E-Learning tools may vary from software developed for the purposes of teaching to general-purpose software, which can be used for sharing presentations over the network or in the class. In particular, Virtual Meeting Tools (VMT) enable real-time interactions through features such as chat tools, audio, video, and user interface screen sharing. They have been used most extensively in education, for example, to arrange remote lectures (ELI, 2006).

Learning in the UML technology context

UML technology supports collaborative R&D processes such as requirements management and software engineering. Kang and Santhanam (2003) identify three knowledge domains that user training should deliver in the context of applications that support collaboration: application knowledge covering commands and tools embedded in the applications; business context knowledge covering the use of IS applications to effectively perform business tasks; and collaborative task knowledge covering how others use the application in their tasks (Table 1).

Users of UML technology need to master all three knowledge domains. They need to understand the business context because UML technology is applicable to many business processes. They need to know collaboration with other users because UML technology mediates collaborative activities in distributed system development. Knowledge of the UML modeling language is a necessary pre-requisite for mastering all three knowledge domains. For example, creating new UML diagrams requires a detailed understanding of the diagram structure. Table 1 illustrates UML technology related knowledge needs.

Previous research on UML technology training

A systematic literature review was conducted (Kitchenham, 2009) to verify to which extent existing studies cover e-Learning usage for UML technology training in industrial settings. We used broad words "training" and "learning" because the e-Learning literature refers to numerous keywords (e.g., online learning, web-based learning, computer-based training, Internet-based training, and web-based training). The review process is as follows:

1. The first criterion was to find UML training related articles by searching words "UML" and "training" or "learning" in the title, abstract, or keywords. Decision was based on the title and the abstract.

2. The second criterion was to categorize research according to whether the research reported industrial experiences or not. Decision was based on the title and the abstract of the article. The content was visited when it was impossible to determine otherwise whether the article reported industrial experiences.

3. The third criterion was to determine whether the research reported experiences related to e-Learning. Decision was based on the title and the abstract of the article. The content was visited when it was impossible to determine otherwise whether the article reported e-Learning related experiences.

Knowledge Domain	Definition	UML knowledge example	UML modeling tool knowledge and skills example
Application Knowledge 1) Command based 2) Tool-procedural 3) Tool-conceptual	1) Commands/ keystrokes needed to execute an operation 2) Knowledge required to combine multiple commands and complete a generic task 3) Knowledge to understand the bigger picture of what to do with a tool	Knowledge of different diagrams Which types of diagrams are related to each other and when it is appropriate to use them?	1) Commands/keystrokes in order to create a UML element 2) Combine multiple commands to complete a UML diagram 3) How the tool facilitates the use of multiple diagrams together?
Business context knowledge 1) Business-procedural 2) Business-motivational	1) How to apply the above levels of knowledge to execute a specific business task? 2) What the tool can do for my job? 3) What is the role of the tool in the organization?	Which diagrams are appropriate for which purposes?	1) Which diagrams to apply and when to support a particular business process (e.g., requirements management)? 2) Which business processes of the organization are supported by the UML tool and why?
Collaborative task knowledge 1) Task interdependencies 2) Collaborative problem solving approach	1) Interdependencies between tasks and their effects upon using a UML tool 2) Collaborative problem solving effort between users	Interdependencies between UML and other modeling languages	1) How tasks completed through the UML tool affect and are affected by other users of the tool (and/or related tools)? 2) Knowledge sharing between users to solve problems

Table 1. The content for training UML technology (Koivulahti-Ojala and Käkölä, 2012).

ACM Portal, IEEE Explore, and ProQuest were searched. They covered IS journals and conferences. Seven articles related to UML technology training in industrial settings were found. Three of them focused on e-Learning. Whenever industrial experiences from UML technology training were reported, the cost effectiveness of training was emphasized (e.g. Anda et al., 2006; Bunse et al., 2006).

Anda et al. (2006) investigated the adoption of UML technology in a global corporation applying a UML-based development method in an international project with 230 system developers, testers and managers.

Adoption was supported by face-to-face training and mentoring. Maximum benefits from UML-based development were not achieved because training was (1) not adapted to the needs of the project and (2) too expensive to provide to people who were not directly involved with UML-based development. Andersson et al. (2010) researched the adoption of UML/SysML modeling principles and tools in an aerospace systems engineering project at Saab Aerosystems. The adoption required a clear strategy including just-in-time face-to-face training and mentor support. These articles did not mention usage of e-Learning tools for UML training. Bunse et al. (2005) studied two enterprises aiming at controlled software development processes that use standardized specification languages such as the UML. First, learning goals and skills were analyzed and face-to-face training sessions were joined in workshops. Online learning and coaching and face-to-face training were then provided. Finally, complex domain-specific exercises had to be solved by the participants in teams.

Bunse et al. (2006) analyzed the design and execution of a UML training program blending e-Learning and face-to-face training in a German corporation. First, the participants worked self-directed with the web-based courseware. This phase was a prerequisite for face-to-face training in the second phase. A several weeks long coaching phase concluded the program. The coach consulted the participants about applying UML in their work. No UML tool related training was included. Koivulahti-Ojala and Käkölä (2012) investigated how a global corporation leveraged VMT for UML technology training of hundreds of employees. However, the use of other e-Learning tools was beyond the scope of their research.

In sum, the extant research of e-Learning practices and tools for UML technology training consists of a few papers covering a limited number of e-Learning tools. Longitudinal studies are missing.

Case study

In case studies researchers find out the conditions of the target organizations by making observations, interviewing, archiving, and recording (Yin, 2013). Benbasat et al. (1987) state three reasons why case studies are suitable for IS research:

- The researcher can study the information system in a natural setting.
- The researcher can answer "how" and "why" questions.
- It is suitable for studying topics in which little formal research has been conducted previously.

All three reasons are valid in this research.

The case organization is a global high-technology corporation, developing products in multiple sites. A commercial UML modeling tool was being rolled out globally to support distributed product development. The name of the tool is confidential. Rollout had started in 2008. Users were working in various sites and different time zones. The number of users was approximately 1700 by the end of 2010 and, after organizational changes, 1300 by the end of 2013. Most users were from the R&D organization. The tool was supported by a virtual team consisting of personnel from the global IT department, the department responsible for process and IS development and support for R&D, and subcontractors working for the departments. The virtual team could decide which media to apply for various purposes within the budget available.

The primary sources of information for the case study were the content created for e-Learning tools by the virtual team and the internal documents (e.g., meeting memos of the virtual team; strategy and plans to improve tool support). Yin (2013) emphasizes the following data collection principles:

1) Prepare a case study database where the raw data and all the data and documents the researcher produces are entered. The case study database included all the material used in the analysis such as screen shots of wikis; entries in Intranet, discussion forums, and training calendars; meeting memos of the virtual team; training materials; draft reports, comments for reports, and final reports; interview plans and interview reports.

2) Use a case study protocol. Additional information was requested from virtual team members in relation to the use of e-Learning tools and the plans and decisions made related to e-Learning practices and tools. The members also reviewed and commented the analysis results.

The first author was previously involved in the UML technology deployment project of the case organization. She holds a managerial position in the department supporting UML technology. The preparation of the case study database and following the use case study protocol were of utmost importance to reduce possible bias caused by the involvement of the first author in the support organization. Second author of this paper reviewed all the phases of the study including the content of case study database and the case study protocol (e.g., conclusions made basis on meeting memos). The managerial position of the first author enabled the authors to access all internal materials, including meeting memos, e-mails, and project documents. As the corporate culture encouraged knowledge sharing, it was possible to receive feedback and deal with open issues easily.

Analysis and findings

Classes of tools applied in the case organization for e-Learning are listed in Table 2. Content of the knowledge shared through the tools was evaluated with respect to the proposed model for UML training (Table 1).

Wiki

Users could access a wiki to share application knowledge (e.g., commercial plug-ins and their installation instructions) and collaborative task knowledge. The virtual team made most updates in the wiki. The usage of the wiki by other stakeholders was modest. Later on, the virtual team abandoned the wiki and placed the same material in Intranet not only due to the modest use but also because the case organization had had three mutually incompatible technical implementations of wikis running in parallel and with varying functionality for several years. It was technically easier to deal with one Intranet site than the incompatible wiki implementations.

Intranet

Users could access Intranet pages for application knowledge (e.g., self-study materials), collaborative task knowledge (e.g., contact information for teams applying UML technology), and business context knowledge (e.g., best practices in terms of business targets fulfilled by leveraging UML technology). The virtual team updated Intranet partly based on discussions and meetings with users. Intranet became increasingly important when the virtual team had to abandon the wiki and the discussion forum due to technical reasons and their modest usage.

Discussion forum

Users had access to a discussion forum to share application knowledge and to solve problems collaboratively. The topics discussed were, for example, questions related to UML technology, feature improvement proposals, and tips and tricks for using UML technology more effectively. The intensity of the discussion forum usage was modest. For example, only eight messages were sent between June 2010 and November 2010. Later on, the virtual team abandoned the discussion forum mainly due to the modest usage.

E-mail

The virtual team used e-mail to share application knowledge with all users. It sent business-critical messages to all users (e.g., concerning maintenance breaks) and other messages (e.g., related to training events) to those who had requested information sharing e-mails. Users did not use the distribution list for sharing collaborative task knowledge and business context knowledge. The number of messages sent within a certain period varied significantly. For example, 33 e-mails were sent during a six-month period (June 2013-November 2013).

Virtual Meeting Tool (VMT)

VMT was applied to share application knowledge, collaborative task knowledge, and business context knowledge. Tens of training sessions were organized and users shared best practices of tool usage with other teams. Application knowledge was shared in training sessions provided in co-operation with the

UML tool vendor. The vendor provided training, consultancy, and technical support for its products globally. Each session lasted 1-2 hours including the time for questions and answers. The sessions covered these subjects: Introduction to UML, Introduction to UML Tool, Class Diagrams, Sequence Diagrams, Composite Structure Diagrams, State Machine Diagrams, Use Case Diagrams, Introduction to Collaboration Capabilities of UML Tool, and Introduction to Publishing Models. All the sessions apart from Introduction to UML and Introduction to UML Tool offered application knowledge from three perspectives: 1) command-based, 2) tool-procedural, and 3) tool-conceptual. For example, in the Class Diagrams session, basics of a UML Class diagram were introduced (tool-conceptual), examples how to create an element in a class diagram were given (command based), and examples how to complete a class diagram were provided (tool-procedural). Business context and collaborative task knowledge were also shared. For example, Introduction to UML Tool communicated the status of UML deployment in the case organization (e.g., in which parts of the organization UML technology had been deployed and for what purposes).

E-Learning tool	Knowledge	Content
Wiki	Application knowledge	Sharing commercial plug-ins and related installation instructions/training materials
	Collaborative task knowledge	Sharing plug-ins made by users and related installation instructions/training materials
Intranet	Application knowledge	Self-study training materials Frequently Asked Questions New features of each tool release Installation instructions How to apply to use the tool Recorded training sessions Material from other sessions
	Collaborative task knowledge	List of contact persons for teams using UML technology Contact information for tool support team Recorded training sessions Material from other sessions
	Business context knowledge	Best practices in the form of business targets, UML modeling conventions, and deployment activities Recorded training sessions Material from other sessions
Discussion forum	Application knowledge	Shared application knowledge
	Collaborative task knowledge	Solving problems collaboratively
E-mail	Application knowledge	Informing all users about maintenance breaks, new features, training and other sessions to be organized
VMT	Application knowledge	Training sessions Sessions where active users shared best practices and application knowledge with other teams about applying the tool
	Collaborative task knowledge	Training sessions Sessions where active users shared best practices with other teams about applying the tool for modeling, including collaborative task knowledge
	Business context knowledge	Training sessions Sessions where active users shared best practices with other teams about applying the tool for modeling, including business context knowledge

Table 2. E-Learning tools the virtual team applied for e-Learning in the case organization.

Trainers were either vendor personnel or specialists of the case organization experienced in both training and VMT. Sessions were organized using standard conference calls and a VMT. Most users had several years of experience in using both conference calls and VMT. During a six-month period (June 2010-

November 2010), 29 1-2 hour-sessions were organized. After three years, VMT was still extensively used. During a six-month period (June 2013-November 2013), 29 1-2 hour-sessions were organized. Users increasingly proficient in UML technology still found it beneficial to join sessions because the sessions supported both novice and advanced learners and the ongoing improvement of ways of working. Users also encountered new business contexts requiring them to use, for example, UML diagrams unfamiliar to them.

Collaborative task and business context knowledge were mainly shared in 2-4 sessions organized in a year for local contact persons responsible for supporting their teams in UML technology usage. For example, sessions offered business context knowledge about on-going UML technology deployment projects and asked users to share their modeling practices (i.e., collaborative task knowledge). In sum, VMT was mostly used for application knowledge training but some business context and collaborative task knowledge were also shared. The virtual team had plans to apply VMT to share business context knowledge more intensively, for example, by communicating business needs and proposals for more advanced ways of using UML technology.

Face-to-face training sessions were also organized. UML technology experts, for example, suggested suitable diagrams, structured the models, and provided tailored training to meet specific modeling needs. Users participated in on-line training sessions before and after face-to-face training sessions, respectively, to get familiar with the subject and to refresh their memories. Surveys, conducted to understand users' backgrounds before training, revealed that users' previous UML technology experiences varied from none to several years.

Evolution of e-Learning over time

Rather than viewing training as an activity confined to coaching users before the deployment of a system, training activity has to extend to coaching users for a period during the actual use of the system (Kang and Santhanam, 2003). Therefore, we extended the analysis to the phase when the UML technology deployment was started and probed future training plans. Table 3 summarizes the e-Learning tools involved in different phases.

The application of e-Learning tools first extended over time in terms of (1) application knowledge, business context knowledge, and collaborative task knowledge and (2) the number of e-Learning tools applied. When the deployment project started, VMT was used only for meeting purposes. During deployment, a user satisfaction survey was organized (Islam et al., 2010). Training, communication, and user guides were improved based on the survey results. VMT was applied for training as well, because users were working in distributed sites and the travelling costs needed to be cut down. All e-Learning tools were used for sharing application knowledge when the deployment started. There are two main reasons for this. First, the deployment was made on a volunteer basis so organizational units were able to decide about UML technology deployment. There was thus limited collaborative task and business context knowledge available to train or share in the early phase of deployment. Second, the virtual team members were mainly experienced in the area of application knowledge sharing. They felt most comfortable to start applying e-Learning tools in application knowledge sharing and training.

Over time e-Learning was used to support also the creation and sharing of business context and collaborative task knowledge. During the analyzed period, the wiki and the discussion forum, applicable to sharing both application knowledge and collaborative task knowledge, were in limited use for those purposes whereas VMT was extensively used for application knowledge training and, to some extent, for sharing collaborative task knowledge and business context knowledge. When the number of UML users started to reduce, the virtual team stopped using the wiki and the discussion forum mainly due to the modest usage. However, the wiki and the discussion forum had been used during the deployment to create and share some highly relevant collaborative task knowledge. The virtual team transferred this knowledge to Intranet. Without the wiki and the discussion forum, this knowledge might never have been created. Therefore, these two classes of tools may be valuable to other organizations even if the use of their instances was stopped during the case study. To deploy UML technology more extensively, VMT was used even more for the training of business context knowledge. During 2013, users were increasingly competent with UML technology and needed little face-to-face training. VMT was the main tool to deliver application, collaborative task, and business context knowledge.

E-Learning tool	When UML technology deployment started (2008)	Status in 2010	Status in 2013
Wiki	Virtual team used it for sharing application knowledge.	Virtual team and users used it for sharing application and collaborative task knowledge.	Wiki was no longer used. Its most relevant knowledge had been transferred to Intranet.
Intranet	Virtual team used it for sharing application knowledge.	Virtual team used it for sharing application knowledge, collaborative task knowledge, and to some extent business context knowledge.	Virtual team extended its use for business context knowledge sharing more extensively.
Discussion forum	Virtual team used it for sharing application knowledge.	Virtual team and users used it for sharing application knowledge and collaborative task knowledge.	Discussion forum was no longer used. Its most relevant knowledge had been transferred to Intranet.
E-mail	Virtual team used it for sharing application knowledge.	Virtual team used it for sharing application knowledge.	Virtual team used it for sharing application knowledge.
VMT	It was only used for meetings.	Virtual team used it for sharing application knowledge, collaborative task knowledge, and to some extent business context knowledge.	Virtual team extended its use for business context knowledge creation (e.g., creation of UML models).

Table 3. E-Learning tools the virtual team applied for e-Learning over time.

Evaluation of the deployed e-Learning tools

Based on the case study, face-to-face training and support were accompanied by a wide variety of e-Learning tools including wiki, discussion forum, Intranet, e-mail, and a VMT. The application of e-Learning tools for software application training focused first on application knowledge training but extended over time to include business context knowledge and collaborative task knowledge. The number of e-Learning tools applied increased for some time and then reduced to a few. In the beginning, the UML tool vendor and the virtual team responsible for the global deployment of UML technology produced most learning content but over time the community using the technology became a more and more prominent content contributor.

The evaluation of the success of e-Learning tools in the case organization was conducted both directly and indirectly. VMT tool use was evaluated in on-line training sessions covering UML technology. According to the study, skills, knowledge, and motivation of users were improved and learners were satisfied with the training (Koivulahti-Ojala and Käkölä, 2012). In addition, two user satisfaction surveys were conducted in 2009 (Islam et al., 2010). The team analyzed the results of the surveys; concluded that instructions, user guides, and training practices had to be improved; and initiated improvement activities during 2009. Table 3 reflects pre-survey and post-survey status. The user satisfaction improved after the VMT was deployed in training and the usage of Intranet was extended to support the sharing of all three types of knowledge.

This study was conducted within one case organization, limiting the generalizability of the results. The analysis also concentrated on the e-Learning content, tools, and practices provided by the virtual team. The community using UML technology may have leveraged other e-Learning content and practices as well but it is beyond the scope of this paper to analyze the content and practices created by several thousands of UML technology users in depth longitudinally.

Conclusions and future research

Little longitudinal scientific research has been conducted to help practitioners to deploy e-Learning practices and tools in industrial settings for teaching and learning complex technologies such as the UML technology. This paper probed through a case study, which classes of e-Learning tools organizations should leverage for organizing and delivering technology training scalably so that large numbers of learners learn application, business context, and collaborative task knowledge in order to master complex technologies. In addition to face-to-face training sessions, the case organization experimented in large scale with instances of several classes of e-Learning tools including wiki; discussion forum; Intranet; e-mail and a VMT. Interestingly, the case organization used most tools to support the application, collaborative task, and business context knowledge learning and sharing as called for by Kang and Santhanam (2003). E-Learning improved user satisfaction with UML technology (Islam et al., 2010). The discussion forum and the wiki proved ineffective over long-term use in comparison with Intranet, email, and VMT and were abandoned. In sum, Intranet, email, and VMT proved the most effective classes of tools for learning application, business context, and collaborative task knowledge. VMT was the most crucial class of tools because the VMT instance not only contributed to the sharing of all three types of knowledge in the case organization but also improved the motivation of the users to use UML technology.

Organizations planning to apply e-Learning for complex technology training should consider the following lessons. 1) Adopting e-Learning tools stakeholders are unfamiliar with requires substantial effort over time. For example, during the case study, the UML tool vendor needed to invest significantly in learning and installing the needed set of tools. Organizations must ensure those planning and sharing content will get familiar with the tools. If an organization already has widely used e-Learning tools, those tools are good candidates for creating and sharing e-Learning content. 2) Face-to-face training and e-Learning need to be planned and delivered in a coordinated fashion especially for sharing collaborative task and business context knowledge. Trainers, e-Learning tool specialists, and tool support personnel need to work together to plan, implement, and deploy e-Learning and face-to-face training. 3) Training sessions need to be organized so the content is meaningful from the users' perspectives and the sessions are short enough to fit most user schedules. In the case organization, the VMT-supported training sessions were running for several years successfully in terms of increased knowledge, skills, and motivation. The sessions were planned especially for the target group, helping learners to reach their learning objectives. 4) Whenever the use of the technology requires application, collaborative task, and business context knowledge, training must enable the sharing of all three types of knowledge. In the case organization, the UML training was extended during the five years of longitudinal study to cover the three types of knowledge.

Future research is needed to determine which combinations of tools and face-to-face training are the best to support the learning of complex technologies in ways that learners are satisfied with the training and the training positively impacts the skills, knowledge, and motivation of the learners. The use of e-Learning tools evolved over time in the case organization with respect to the number of tools deployed and the coverage of not only the application knowledge training but also the business context knowledge and collaborative task knowledge training. The usage of various classes of e-Learning tools should be studied in future longitudinal studies to understand these phenomena better. For example, the classes of wiki and discussion forum tools need to be investigated further because during this study they were initially found useful but eventually abandoned. Several theoretical lenses are applicable to understand better the evolution of e-Learning tool use over time. The innovation diffusion theory is a possible lens because e-Learning practices and tools can be seen as a technology, which is adopted and reconstituted over time. The diffusion of technology can be conceptualized in terms of organizational learning, skill development, and knowledge barriers (Attewell, 1992). Application of diffusion theories in studies of e-Learning could provide researchers and practitioners with a better understanding to successfully adopt e-Learning in industrial settings.

References

Anda, B., Hansen, K., Gullesen, I., and Thorsen, H. 2006. "Experiences from introducing UML-based development in a large safety-critical project," *Empirical Software Engineering* (11:4), pp. 555-581.

Andersson, H., Herzog, E., Johansson, G., and Johansson, O. 2010. "Experience from introducing unified modeling language/systems modeling language at Saab Aerosystems," *Systems Engineering* (13:4), pp. 369-380.

Attewell, P. 1992. "Technology diffusion and organizational learning: The case of business computing," *Organization Science* (3:1), pp. 1-19.

Benbasat, I., Goldstein, D. K., and Mead, M. 1987. "The case research strategy in studies of information systems," *MIS Quarterly* (11:3), pp. 369-386.

Bostrom, R., Olfman, L., and Sein, M. 1990." The importance of learning style in end-user training," *MIS Quarterly* (14:1), pp. 101-119.

Bunse, C., Grutzner, I., Peper, C., Ochs, M., and Peper, C. 2005. "Applying a Blended Learning Strategy for Software Engineering Education," in *Proceedings of 18th Conference on Software Engineering Education and Training*, IEEE, pp. 95-120.

Bunse, C., Grutzner, I., Peper, C., Steinbach-Nordmann, S., and Vollmers, C. 2006. "Coaching professional software developers - an experience report," in *Proceedings of 19th Conference on Software Engineering Education and Training*, IEEE, pp. 123-130.

Compeau, D., Olfman, L., Sei, M., and Webster, J. 1995. "End-user training and learning," *Communications of the ACM* (38:7), pp. 24-26.

Dori, D. 2002. "Why significant UML change is unlikely," *Communications of the ACM* (45:11), pp. 82-85.

ELI. 2006. "7 things you should know about virtual meetings," Educause (http://www.educause.edu/library/resources/7-things-you-should-know-about-virtual-meetings; accessed April 15, 2014).

Hrastinski, S. 2008. "Asynchronous and Synchronous E-learning," EDUCAUSE Quarterly, (31:4), pp. 51-55.

Islam, A.K.M.N., Koivulahti-Ojala, M., and Käkölä, T. 2010. "A lightweight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool," in *Proceedings of the 16th Americas Conference on Information Systems (AMCIS)*, Paper 287.

Kang, D., and Santhanam, R. 2003. "A longitudinal field study of training practices in a collaborative application environment," *Journal of Management Information Systems* (20:3), pp. 257-281.

Kitchenham, B., Brereton, P., O., Budgen, D., Turner, M., Bailey, J., and Linkman, S. 2009. "Systematic literature reviews in software engineering-A systematic literature review," *Information and Software Technology* (51:1), pp. 7-15.

Kobryn, C. 2002. "Will UML 2.0 be agile or awkward?" *Communications of the ACM* (45:1), pp. 107-110.

Koivulahti-Ojala, M., and Käkölä, T. 2012. "Design, implementation, and evaluation of a Virtual Meeting Tool-based innovation for UML technology training in global organizations," in *Proceedings of 45th Conference on System Science (HICSS)*, IEEE, pp. 3980-3989.

Larman, C. 2005. *Applying UML and Patterns: An introduction to the object-oriented analysis and design and iterative development (*3rd ed.), Pearson Education.

Nelson, R., Whitener, E., and Philcox, H. 1995. "The assessment of end-user training needs," *Communications of the ACM* (38:7), pp. 27-39.

Tavangarian, D., Leypold, M., Nölting, K., Röser, M., and Voigt, D. 2004. "Is e-Learning the Solution for Individual Learning," *Electronic Journal of e-Learning* (2:2), pp. 273-280.

ISO. 2012. *ISO/IEC 19505-1:2012 - Information technology - Object Management Group Unified Modeling Language (OMG UML) - Part 1: Infrastructure*, Geneva, Switzerland: International Organization for Standardization.

Yin, R. K. 2013. *Case study research: Design and methods* (5th ed.), Beverly Hills, CA: Sage Publishing.