Changing the planning for agile and lean software development

From roadmapping to continuous planning

Tanja Suomalainen

VTT SCIENCE 132

Changing the planning for agile and lean software development

From roadmapping to continuous planning

Tanja Suomalainen

Thesis for the degree of Doctor of Philosophy to be presented with due permission for public examination and criticism in auditorium IT116, at the University of Oulu, Linnanmaa, on the 9th of September 2016, at 12 noon.

ISBN 978-951-38-8446-8 (Soft back ed.)
ISBN 978-951-38-8445-1 (URL: http://www.vttresearch.com/impact/publications)

VTT Science 132

ISSN-L 2242-119X
ISSN 2242-119X (Print)
ISSN 2242-1203 (Online)
http://urn.fi/URN:ISBN:978-951-38-8445-1

JULKAISIJA – UTGIVARE – PUBLISHER

Teknologian tutkimuskeskus VTT Oy
PL 1000 (Tekniikantie 4 A, Espoo)
02044 VTT
Puh. 020 722 111, faksi 020 722 7001

Teknologiska forskningscentralen VTT Ab
PB 1000 (Teknikvägen 4 A, Esbo)
FI-02044 VTT
Tfn +358 20 722 111, telefax +358 20 722 7001

VTT Technical Research Centre of Finland Ltd
P.O. Box 1000 (Tekniikantie 4 A, Espoo)
FI-02044 VTT, Finland
Tel. +358 20 722 111, fax +358 20 722 7001

Juvenes Print, Tampere 2016

Preface

My dream of becoming a doctor of philosophy (PhD) was born many years ago. Already while writing my Master's thesis, I realised that this is something that I wanted to do. The PhD journey has required both my passion and persistence, but there have also been a lot of people to thank for making my dream of a completing a PhD a reality.

First of all, I would like to thank my principal supervisor, Professor Jouni Similä, who has over the years encouraged me to pursue the dream and believed that I would finalise this thesis one day. Thank you also for the valuable guidance and comments on the studies, papers, and the thesis itself. Secondly, I would like to thank my other supervisor, Professor Veikko Seppänen, for bringing speed and focus to the actual writing work. I am also really grateful for the endless discussions about drawing pictures and improving the content of the thesis, as well as your extremely fast replies to all of my questions. Thirdly, I want to thank Professor Pekka Abrahamsson for the conversations and advice that you have given to me along the way, which have calmed my mind in problematic situations.

I would also like to thank the reviewers of this thesis, Professor Casper Lassenius of Aalto University, Finland, and Professor Pasi Tyrväinen of the University of Jyväskylä, Finland. I express my sincere thanks for the time and effort they have spent in reviewing my research and giving their constructive comments and recommendations, which have helped me to improve the quality of the thesis.

I am also grateful to VTT for giving me the opportunity to work on various projects to collect the data for my thesis, as well as to write and thereafter publish the work. I am grateful to Dr Tua Huomo, who took me into the Cloud software program and gave me valuable advice at the beginning of the writing process. Also, I want to thank Dr Raija Kuusela for taking me to the N4S research programme and letting me work freely on my thesis while still believing that I could handle it on my own. I really appreciate that you have always been there for me, listening my troubles and excitement along the way and giving me guidance and support when needed. Furthermore, as this thesis is based on five papers, I wish to thank all my co-authors for their contributions. Especial thanks to colleague, co-author, friend and aunt Dr Maarit Tihinen for your friendship, help, and encouragement throughout. Also, I want to thank my longitudinal room-mate and

colleague Susanna Teppola for your support and putting up with me over the years. I also, I want to thank my colleague Kaisa Koskela-Huotari for helping me to organise some of the interviews from one of the case companies.

Over the years, I have worked with many great people, both at VTT and in the case companies, and I wish to thank all of you. Since this thesis would not have been possible without the case companies involved, I offer all the case company representatives as well as all the interviewees who are part of this research much gratitude. Furthermore, there have been so many talented and great colleagues during my career who I have been privileged to work with, but as you are so many, it would be impossible to name you all here. I hope you will understand.

Then to my loved ones, I wish to express my sincere gratitude for your loving support, understanding, and encouragement through the years. First of all, I want to thank my loving husband Lari Suomalainen just for being there for me, and loving me from the bottom of your heart. You have always tried to make my dreams come true and helped me achieve my goals. I am grateful that you understand me and my temper as well as changes in my mood so well. Then, thanks to my loveable children: Veera, Luka, and Viola, for bringing so much love and joy to my life. I also want to thank my mum and dad for their endless love and support in all areas of life. I also want to thank my mother-in-law Irma Suomalainen for taking care of my kids while I needed to work long hours, and my grandma Rauha Oikarinen for giving all the best instructions for life and providing me with the time to relax when I have needed it. Finally, I want to thank all my friends who have supported and encouraged me along the way. I want to thank my support group ‘baby greetings’, for understanding me as a woman and a mum and empathising the small and large joys and sorrows of life. In all, this would not have become true without all of you.

Tanja Suomalainen
Oulu, Finland, August 2016

Academic dissertation

Supervisors

Professor Jouni Similä
University of Oulu
Faculty of Information Technology and Electrical Engineering
Department of Information Processing Science
P.O. Box 3000, 90014 University of Oulu, Finland

Professor Veikko Seppänen
University of Oulu
Martti Ahtisaari Institute, Oulu Business School, and
Faculty of Information Technology and Electrical Engineering
P.O. Box 3000, 90014 University of Oulu, Finland

Reviewers

Professor Casper Lassenius
Aalto University
Department of Computer Science
P.O. Box 15400, 00076 Aalto, Finland

Professor Pasi Tyrväinen
University of Jyväskylä
Department of Computer Science and Information Systems
P.O. Box 35, 40014 Jyväskylä, Finland

Opponent

Professor Hannu Jaakkola
Tampere University of Technology
Department of Software Engineering
P.O. Box 300, 28101 Pori, Finland

List of original publications

This thesis is based on the following original publications which are referred to in the text as Papers I–V. The publications are reproduced with kind permission of the publishers.

I Suomalainen, T., Tihinen, M., and Parviainen, P. (2009). *Challenges for Product Roadmapping in Inter-company Collaboration*. In: Proceedings of the Third International Conference on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD). ETH Zurich, Switzerland on July 2–3, 2009. Springer, pp. 66–80.

II Suomalainen, T., Salo, O., Abrahamsson, P., and Similä, J. (2011). *Software Product Roadmapping in a Volatile Business Environment*. The Journal of Systems and Software, Vol. 84, Issue 6, pp. 958–975.

III Suomalainen, T., Kuusela, R., and Tihinen, M. (2015). *Continuous Planning: An Important aspect of Agile and Lean Development*. International Journal of Agile Systems and Management (IJASM), Vol. 8, No. 2, pp. 132–162.

IV Suomalainen, T. (2015). *Defining Continuous Planning through a Multiple-Case Study*. In: Proceedings of the 16th International Conference of Product-Focused Software Process Improvement (PROFES). Bolzano, Italy on December 2–4, 2015. Springer, LNCS 9459, pp. 288–294.

V Rodrıguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S, Suomalainen, T., Eskeli, J., Karvonen, T., Kuvaja, P., Verner, J. M., and Oivo, M. (2016). *Continuous Deployment of Software Intensive Products and Services: A Systematic Mapping Study.* The Journal of Systems and Software, in press.

Author's contributions

Paper I '*Challenges for product roadmapping in inter-company collaboration*' describes product roadmapping from the inter-company collaboration perspective. Collaboration is defined as an activity in which two or more parties, e.g. companies, departments, customers, or agencies work together to create a mutual value and achieve a common goal. The paper discusses the challenges and opportunities that inter-company collaboration (that is joint R&D partnerships, customer–supplier relationships, and technology exchange agreements and licencing) sets for product roadmapping, including, for example, the most important activities to consider, the most typical problems and how those problems can be avoided. The empirical research was conducted as a questionnaire study and interviews, which were planned, carried out, and analysed by Suomalainen. She was also responsible for the planning and execution of related research and was the main author of the *conference paper*.

Paper II '*Software product roadmapping in a volatile business environment*' develops a framework for software product roadmapping, which is used to study the critical aspects of the product roadmapping process. The research findings presented in the paper aim at defining product roadmapping based on the current literature and empirical research, including an overview of roadmapping, stakeholders of the product roadmapping process and product roadmapping phases. The empirical research was conducted as a questionnaire study and interviews. Both the empirical and scientific part of the research were planned, executed, and analysed by Suomalainen. Suomalainen was the main author of the *journal paper.*

Paper III '*Continuous planning: an important aspect of agile and lean development*' defines continuous planning though a multiple-case study. The paper highlights the importance of continuous planning throughout an entire organisation, including the elements of continuous planning (that is organisational planning, strategic planning, and business planning) and their close interrelation. Organisational planning serves to define a plan's organisational level and the time frames of a plan, strategic planning serves to set an overall plan of an organisation, and business planning serves to establish the budgeting frame of a plan. The empirical evidence of the paper is drawn from the experiences of three

case companies in terms of how they viewed and conducted continuous planning. The author was responsible for the planning and execution of related research. Also, the author conducted and analysed the case studies relating to cases A and B. Raija Kuusela conducted and analysed the case study relating to case C. Suomalainen was the main author of the *journal paper*, and Kuusela and Tihinen provided comments and feedback to the manuscript.

Paper IV '*Defining continuous planning through a multiple-case study*' presents the results of a multiple-case study in which the different levels of planning (that is strategic, financial, business, product, and team) along with their time frames are explored. The research findings presented in the paper reveal the practices of continuous planning among the case companies with their key activities. The empirical evidence of the paper is drawn from the experiences of three case companies, which were collected through several interviews. Suomalainen was responsible for planning, carrying out, and analysing the interviews. Suomalainen was also responsible for the planning and execution of the related work and she was the sole author of the *conference paper.*

Paper V '*Continuous deployment of software intensive products and services: a systematic mapping study*' defines the method and the main findings of a systematic mapping study about continuous deployment. The paper classified and analysed the literature related to continuous deployment in the software domain in order to scope the phenomenon, provided an overview of its state-of-the-art, investigated the scientific evidence in the reported results, and identified areas that were suitable for further research. Suomalainen was one of the participants of the research group conducting the systematic mapping study of the continuous deployment. During the analysis of the research results, factors relating to continuous deployment were divided among the researchers of the research group based on their interests and research topics. Suomalainen was responsible for the factors relating to fast and frequent release, including also continuous planning, as well as agile and lean software development in the continuous deployment context. Suomalainen was a co-author of the paper as the *journal paper* was written together by the research group.

Contents

List of abbreviations

AHP	Analytical Hierarchy Process
ART	Agile Release Train
ASD	Agile Software Development
ASDE	Agile Software Development Ecosystem
CD	Continuous Deployment
CP	Continuous Planning
CPEF	Continuous Planning and Execution Framework
CSO	Chief Strategy Officer
CTO	Chief Technology Officer
ICT	Information and Communication Technology
ID	Identification
IoT	Internet of Things
IT	Information Technology
ITEA	Information Technology for European Advancement
JIT	Just-In-Time
LD	Lean Development
MERLIN	eMbedded Systems Engineering in Collaboration
MMF	Minimum Marketable Features
N4S	Need for Speed
PhD	Doctor of Philosophy
PSI	Potentially Shippable Increment
QA	Quality Assurance

QFD	Quality Function Deployment
R&D	Research and Development
RE	Requirements Engineering
ROI	Return on Investment
RQ	Research Questions
SE	Software Engineering
SLR	Systematic Literature Review
VTT	VTT Technical Research Centre of Finland Ltd
XP	eXtreme Programming

1. Introduction

Market uncertainties, competitive pressures and the constant need for shortened development cycles call for flexible, responsive and adaptive software development practices (Olsson et al. 2013). Since the mid-1990s, a variety of agile methods and practices have been designed so as to enhance development teams' or an organisation's ability to respond to dynamic market changes (Highsmith 2002a, Kettunen 2009, Olsson et al. 2013). Also, in recent years, lean thinking (e.g. Womack and Jones 2003) has been introduced in software development companies (Poppendieck and Poppendieck 2003, Charette 2003, Middleton et al. 2005) with the aim of achieving a continuous and smooth flow of production in pursuance of removing waste in processes. The promise of lean development is to create a change-tolerant organisation that can survive and succeed in times of uncertainty, change and complexity (Charette 2003). In emphasising the use of iterations and the development of small features, it is clear that agile practices and lean development have indeed increased the ability for software development companies to accommodate fast-changing customer requirements and fluctuating market needs, as well as having reduced lead times and improved the quality of their products (Olsson et al. 2013).

Even though many software development companies have succeeded in adopting agile practices in order to improve responsiveness to customers, agile development is not the end or final step of software development (Olsson et al. 2013). New and innovative approaches that support continuous practices are needed. In fact, it has been realised that software development companies are moving towards continuous deployment (CD) in which software functionality is continuously deployed and where customer feedback is the main driver for innovation (Olsson et al. 2013). CD refers to the organisational capability to develop, release and learn in rapid parallel cycles, such as in hours, days or a very few weeks and turning the software development into a continuous flow. According to Järvinen et al. (2014), CD refers to the ability to deliver the smallest added value to the customers, which requires automating all the processes that must be executed to deliver the software to customers, and thus implementing CD concerns the whole company. Similarly, Fitzgerald and Stol (2014) emphasize a closer connection between development and execution in order to detect and fix errors as soon as possible. Accordingly, the integration between software

development and its operational deployment should be continuous. Similarly, the link between business strategy and software development should be continuously assessed and improved (Fitzgerald and Stol 2014).

All of these software development practices – agile, lean, and continuous deployment – change, among other things, the scheduling and planning. Planning in general is seen as consisting of both actions and forecasts (that is the expected outcomes). Roadmapping is a planning process for creating and revising future plans, that is roadmaps (Kostoff and Schaller 2001). Roadmaps are also forecasts of what is possible or likely to happen, and plans that express a course of action (Kappel 2001). A roadmap structure consists of both layers and time frames. The structure is flexible, and can be adapted to address many issues at different levels of an organisation (Phaal and Muller 2009). In large organisations, there are multiple levels of planning performed in different time frames and by different actors (e.g. Cohn 2006, Leffingwell 2011, Kuusela and Koivuluoma 2011). Even though roadmapping can be used as a stand-alone problem-solving method, it will have more impact if it is integrated with the core business processes where decisions are made and budgets allocated (Cosner et al. 2007). In the software development context, planning is commonly episodic and performed according to a traditional cycle, usually triggered by annual financial year-end considerations (Fitzgerald and Stol 2014). In fact, the problem in planning is that time is divided into a number of planning horizons, each lasting a significant period of time, and continuity is not seen throughout the organisation. Yet today, as agile and lean practices are becoming the norm, the transformation towards continuous practices is being emphasised in organisations. Drawing on the lean concept of flow, a number of continuous activities are identified (Fitzgerald and Stol 2014) that are important for software development in today's context, one of them being continuous planning.

Continuous planning is about implementing planning practices continuously, not just as part of a top-down annual event (Hope and Fraser 2003). Environmental changes trigger continuous planning instead of the predefined and regular planning cycle (for example, the financial year), and thus plans are adjusted according to internal and external events (Rickards and Ritsert 2012). Planning should be carried out continuously so that, at any time, the full scale of the development can be presented (Westkamper and Von Briel 2001). Fitzgerald and Stol (2014) define continuous planning as a holistic attempt involving multiple stakeholders from business and software functions, whereby plans are dynamic open-ended artefacts that evolve in response to changes in the business environment, and thus involve a tighter integration between planning and execution. In software development, continuous planning refers to the organisational capability to conduct planning in rapid parallel cycles, which can be hours, days or very small numbers of weeks or months, depending on the level of planning.

1.1 Background and motivation

The research started already in 2006 with the topic of roadmapping. Back then, the literature on how roadmapping is conducted in a software development context was not extensive. Most of the literature was written from the perspective of industry, science or technology roadmapping, not from the perspective of product roadmapping involving such aspects as releases, features, requirements or other even smaller components of software product planning. However, interest in roadmapping has increased a great deal since then, and it has been noticed more widely both in research as well as in industry.

Roadmapping, which was initially developed in the 1970s at Motorola to improve the alignment between technology and innovation (Willyard and McCless 1987), has now spread to several other industrial sectors, for example, from medical care to software development. It has evolved and become popular in the last decade, as it has been adopted by companies, governments, and other institutions (Phaal et al. 2008, Phaal et al. 2009). It has even been claimed that roadmapping is one of the most widely used planning practice and management techniques for supporting innovation and strategy (Phaal and Muller 2009). It is also anticipated that roadmapping will increasingly be used to support strategic dialogue in the future. However, even though it is claimed (e.g. Kostoff and Schaller 2001, Lee and Park 2005) that roadmapping is popular and widespread, Carcalho et al. (2013) point out that there is still little or no evidence from localized surveys of management practices to support these assertions, and thus, further research is needed in that regard. Also, based on the current literature (e.g. Phaal and Muller 2009), the development of roadmapping has largely been driven by practice (within companies, government agencies and consulting firms), with relatively little academic research to support the theoretical underpinning of the method. Therefore, academic research is also relevant.

Furthermore, roadmapping has not been widely studied from the software development perspective, or more precisely from the agile and lean software development perspective, which would complement the other research areas in the field of research. Also, how roadmapping should be integrated with other management processes, methods and frameworks is not well understood. Research is needed to determine how organisations have aligned roadmapping with their core business processes (for example, strategic, financial and business planning and new product introduction), and to describe how the roadmap structure can be applied throughout the organisation.

Unlike roadmapping, continuous planning is a relatively new and hence poorly studied field of research, especially in the software development context. Therefore, the literature relating to continuous planning is not adequate. Based on the current literature, continuous planning is not commonly adopted and applied throughout the organisation and currently involves only a certain level of planning, for example release planning (by using Scrum for instance). In that regard, Fitzgerald and Stol (2014) conclude that continuous planning is not widespread

throughout the organisation. Accordingly, the only form of continuous planning is what emerges from agile development approaches, and is related to sprint iterations or at best, software releases. However, continuous planning requires a wider perspective than is currently considered, including also other levels of planning than just the team-level release planning. Actually, there is not any empirical evidence of how continuous planning is conducted throughout the organisation at the various levels of planning that would also describe the interrelationships between these various levels. The interrelationships between the plans include such information as how the planning as well as the plans are visible to the other levels of planning.

The planning practices in the field of software development are commonly adjusted according to the prevalent practices of software development. In the last ten years, the practices of software development have changed from more traditional, waterfall practices to agile and lean practices, and now even further to continuous deployment. Similarly, planning practices have changed from roadmap-based planning to continuous planning. New planning practices are typically first applied in product planning and thereafter spread and adopted to the other levels of planning in the organisation. All in all, the change in planning has not been systematically explored in the related literature on software development, and what kind of effects it has for agile and lean software development. Also, neither the reasons for the change have been identified nor what kind of effects the changes have.

1.2 Research questions and scope

This subchapter presents the research problem and questions of this thesis. The main research problem is further elaborated by three additional sub-questions, each of them targeting a specific area of interest that contributes to the main research problem. The main research problem is posed as follows:

How has planning changed for agile and lean software development from roadmapping to continuous planning?

In order to provide an answer to this problem, the following research questions (RQ) are defined that approach the main research problem from different perspectives, namely roadmapping and continuous planning, during a period of time in software development:

RQ1. How is roadmapping conducted in software development?
RQ2. How is continuous planning conducted through roadmapping in agile and lean software development?
RQ3. How is continuous planning conducted in continuous deployment-driven software development?

The scope of the research consists of two main topical areas, roadmapping and continuous planning, as illustrated in Figure 1. These two main research topics are studied in the context of software development, involving companies operating in the field of Information and Communication Technology (ICT). It has been realised that both planning and software development practices have changed in recent years. The background for this change lies in agile and lean development methods and practices that have triggered the need both for continuous deployment and continuous planning. Thus, agile and lean software development form the overall research context to this study. This research context has a strong link to lean thinking, as the concept of continuity in software development comes from the lean approach, called flow. Thus, while moving towards continuous deployment, the purpose is to establish a continuous movement, which closely resembles the concept of flow.

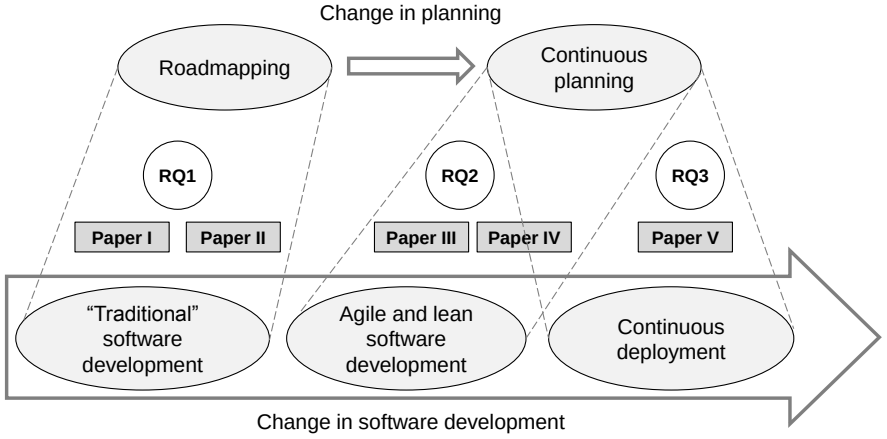


Figure 1. Scope of the research.

As illustrated in Figure 1, this thesis consists of five original papers published in the main topical areas of the research, roadmapping and continuous planning. Each paper in this study tackles the research topic from a particular viewpoint. The figure also presents the relationship between the research questions and publications. The first research question is answered in Papers I and II based on empirical evidence, in which data are gathered through initial inquiry (presented in Chapter 3.4.2). The second research question is answered in Papers III and IV based on empirical evidence, in which data are gathered through a multiple-case study (presented in Chapter 3.4.3). The third research question is answered in Paper V, based on a wide systematic mapping study (presented in Chapter 3.4.1). This theoretical evidence is sufficient at this point in the research, as companies have not yet fully adopted the practices of continuous deployment, while the

purpose is to bring out the future directions for software development and the research at hand.

1.3 Structure of the thesis

The structure of the thesis consists of five chapters including the introduction, theory, research design, original publications, and discussion and conclusions. The thesis is structured as follows.

Chapter 1 is the introduction to this research, presenting the background and focus of the thesis as well as the field of research. It also defines the research questions that will be answered.

Chapter 2 describes the theoretical background of the study. It presents the related work concerning the main areas of the research: 1) agile and lean software development, 2) roadmapping, and 3) continuous planning. Finally, at the end of the chapter, the analysis of the related work is summarised.

Chapter 3 presents the research design of this thesis. It defines the research approach and methods employed. It also introduces the case companies together with the case selection criteria and description. Thereafter, the data collection methods are presented, and different phases of the research are introduced, that is, the initial inquiry and multiple-case studies. Finally, at the end of the chapter, data analysis methods are defined and discussed.

Chapter 4 introduces the original publications of the thesis. It presents the overall view of each paper and identifies the key contributions of each paper in regard to this thesis. It also describes how each publication answers or clarifies the research questions of the thesis.

Chapter 5 discusses the results of the thesis as well as setting out the conclusions. Firstly, answers to the defined research questions are provided in a summary. Secondly, the research results are evaluated. Both the theoretical and managerial implications are addressed, and the reliability and validity of the research is discussed. Also, the limitations of the research are described. Finally, at the end of this chapter, some concluding words are given and the future research opportunities are presented.

2. Background and related work

The aim of this chapter is to provide an overview of the main areas of the research. Firstly, the background of agile and lean software development is presented. This provides the overall context for this research, as it has been realised that agile and lean development practices have triggered the need to change current planning practices. Secondly, roadmapping is described. This involves defining of the roadmap structure as well as describing the process and the participants of the roadmapping activity, especially from the software development perspective. Thirdly, continuous planning is presented. This involves several levels of planning, as presented in the roadmap structure, which are described in more detail from the agile–lean organisation's point of view. Finally, at the end of this chapter, the background and the view of the related work are summarised.

2.1 Agile and lean software development

Given that the current business environment of ICT organisations is very unstable and constantly changing, organisations are increasingly adopting agile and lean development practices. That is because being capable of sensing and responding to predictable and unpredictable events in times of change and uncertainty has become crucial for companies (Baskerville et al. 2005). Agility is about changing businesses and business processes, while sensing environmental changes and responding appropriately (Overby et al. 2005, Van Oosterhout et al. 2005). Lean development, on the other hand, is about creating a change-tolerant organisation that can survive and succeed in times of uncertainty, change, and complexity (Charette 2003).

Agile and lean software development practices are used for both small projects and large development projects with over 100 designers. The trend that large companies are adopting these practices to develop large software products in an iterative manner is increasing (Staron et al. 2012). Similarly, Bellomo et al. (2013) describe that both industry and government have been increasingly adopting agile-based incremental software development practices. For the pervasive adoption of these practices, Papatheocharous and Andreou (2014) provide

evidence-based information from the software and ICT industry regarding the state of practice of agile development. On the other hand, Kurapati et al.'s (2012) research provides evidence on which of these practices are most commonly used at a project and organisational level. There are in fact multiple benefits that are gained though adopting these practices, the most significant being the enhancement of the acceleration of the time to market and the ability to manage and prioritise changing requirements (Papatheocharous and Andreou 2014). These practices also show promise in improving speed and ways to achieve substantial cost savings and quality improvements (Radnor and Walley 2008, Bellomo et al. 2013).

Both of these practices, agile and lean, are seen to complement each other. Petersen (2010) compared these two practices and identified twenty-two agile practices and four lean practices. The practices were then compared so as to highlight the differences and similarities between lean and agile software development. Petersen concluded by saying that 1) both practices share the same goals (focus on the customer), 2) both practices define similar principles, with the principle 'see the whole' being unique to lean, 3) both practices have unique and shared principles, and 4) lean does not define processes while agile defines different ones. The key difference between agile and lean is that, where agile software focuses on the software development function, lean thinking focuses on the end-to-end process: from customer to delivery (Petersen 2010).

2.1.1 Agile software development

Agility is about being able to respond rapidly to unpredictable changes in demand (Holmqvist and Pessi 2005). Furthermore, agility has been defined as the ability to both create and respond to change in order to profit in a turbulent business environment (Highsmith 2002a). Highsmith (2002a) explains that agile organisations foster change by being better than their competitors at responding to changing conditions by creating change that competitors cannot respond to adequately. Accordingly, agile organisations are also nimble (able to change direction quickly) and flexible (able to see how things that worked for them earlier may not work as well in the future). An agile organisation also needs to know how to balance structure and flexibility (Highsmith 2002a).

Agile software development (ASD) is characterised by a chaordic (including chaos and order) perspective, collaborative values and principles, and a barely sufficient methodology. Highsmith (2002b) calls this definition an agile ecosystem. The core set of agile software development ecosystems (ASDE) family includes lean development (LD), Scrum, the dynamic software development method, crystal methods, feature-driven development, extreme programming, and adaptive software development. To understand ASDE, Highsmith (2002b) clarifies: 'Viewing organisations as chaordic means understanding that the predictability upon which traditional project management and development life cycle practices are built is a root cause of dysfunctionality between customer, management, and development

organisations.' A chaordic perspective also impacts how changes are responded to and how project teams and organisations are managed. In everyday work, such a perspective may create two outcomes: product goals are achievable, but they are not predictable, or processes can aid consistency, but they are not repeatable (Highsmith 2002b). According to Ivachtchouk (2004), agile software development emphasises customer satisfaction through a continuous delivery of functional software and accepts changes to requirements in response to evolving business needs. Thus, the agile development approach primarily addresses the problems of rapid change (Ivachtchouk 2004).

The following practices are typical of agile software development: short iterations, continuous testing, self-organizing teams, constant collaboration, and frequent planning based on current reality (Highsmith 2002b). Values and preferences of the agile software development are defined as follows by Agile Manifesto (2001): individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.

Business agility, unlike software agility, is the organisation's ability to sense environmental change and respond appropriately (Evans 2002, Overby et al. 2005). According to Van Oosterhout et al. (2005), it is about being able to quickly and easily change businesses and business processes outside the normal level of flexibility in order to effectively deal with highly unpredictable external and internal changes. Accordingly, the external changes are the consequence of the following domains: social or legal, the business network, the competitive environment, changes in customer needs, and technology. Instead, the internal changes are the consequence of the following: performance indicators, information technology, and mergers and acquisitions. Furthermore, internal changes are those such as a new strategy or a takeover that require the organisation to adapt (Van Oosterhout et al. 2005). According to Overby et al. (2005), an environmental change includes, for instance, changes predicted by competitors' actions, consumer preference changes, regulatory or legal changes, economic shifts, and technological advancements. An appropriate response supports the organisation's goals, such as to increase market share, capture new customers, or avoid competition. The appropriateness of the response is also affected by relative cost and quality (Overby et al. 2005).

The intention of business agility is to be able to create new processes and new value propositions in a rapidly changing business environment and to be able to implement these changes rapidly (Evans 2002). According to Evans (2002), it is a a top-down approach when speed and flexibility are integrated into an organisation's culture and style. However, business agility is not merely about speed; it is also about doing purposeful work in alignment with corporate objectives (Evans 2002). It has three main enabling elements: knowledge management, value-propositioning skills, and response ability (Dove 2005). Accordingly, the focused knowledge management process enables accurate and timely awareness to make a change. Value-propositioning skills enable effective prioritisation and choice-making among competing response-alternatives.

Response ability means ability to change business processes and to customize operational responses in real time (Dove 2005).

2.1.2 Lean software development

Lean thinking can be traced as far back as the beginning of the 1900s, and after that it was strengthened by lean manufacturing in the 1950s (Womack and Jones 2003). According to Charette (2003), lean manufacturing was pioneered over the past 50 years by Japanese automotive manufactures, for example, at Toyota and Honda. The focus in lean manufacturing was on eliminating waste by just-in-time (JIT) production, and the lean process withdraws only the number of parts needed when they were needed (Dittrich et al. 2005). At the end of the 1990s, Womack and Jones (2003) widened the scope of lean thinking from lean manufacturing to the lean enterprise. Later, lean thinking has also been applied in software development (e.g. Charette 2003, Middleton and Sutton 2005, Poppendieck and Poppendieck 2007). Thus, lean software development and its principles are derived from lean manufacturing and thinking (Shalloway et al. 2009).

From the software development perspective, lean development (LD) is a combination of a system of practices, principles, and philosophies for building software systems for a customer's use (Charette 2003). LD is characterised by lean production resources and management of risks by viewing them as an opportunity (Conn 2004). LD addresses issues from a more strategic and top-down fashion than most of the agile approaches that use a bottom-up mode (Charette 2003). It is also said to be the most strategy-intensive of the ASDE family. Charette (2003) explains that an organisation can change its long-term behaviour only with strong commitment from the top, and this commitment should be tied to a vision of what the organisation should be. For LD to be effective, it requires a resource commitment, an allocation of both time and money, from the highest levels of the organisation. Philosophical commitment is also required, because it creates the vision of how the organisation will operate (Charette 2003). The objective behind LD is to create a change-tolerant organisation or an organisation that can survive and succeed at times of uncertainty, change and complexity, and an organisation that uses information technology (IT) to accomplish this (Charette 2003). Accordingly, the organisation's strategy should be characterised as becoming a risk-entrepreneur, which means creating a business philosophy that seeks risks as the currency of profit. Thus, the organisation should know how, when, where, and why to take risks. A good risk-entrepreneur is characterised by the knowledge of how to recognise and exploit the low-risk, high opportunity situations that others miss. Charette (2003) continues: 'LD is most effective in a risk-entrepreneurial organisation'.

According to Charette (2003), the goal of LD is to create software-intensive systems of size in one-third of the time, at one-third of the cost, and with one-third of the defect rate of more traditional software development approaches. The intention is to select the proper mix of processes, methods, and tools to develop

change-tolerant software within the specified constraints. It also requires organisational support to make everything come together. The main idea behind LD is not only to increase productivity, but also to eliminate waste, for example waste of time, information, and resources. An organisation can eliminate substantial development time and costs by gaining a better understanding of a customer's genuine needs (Charette 2003). Similarly, Liker (2004) presents 14 lean principles, which are divided according to four categories: philosophy, process, people and partners, and problem solving, as shown in Figure 2. These principles have been used as grounds for lean software development. For example, for software development according to Shalloway et al. (2009), lean provides the following principles: respect people, eliminate waste, defer commitment, create knowledge, deliver fast, build quality in, and optimize the whole.

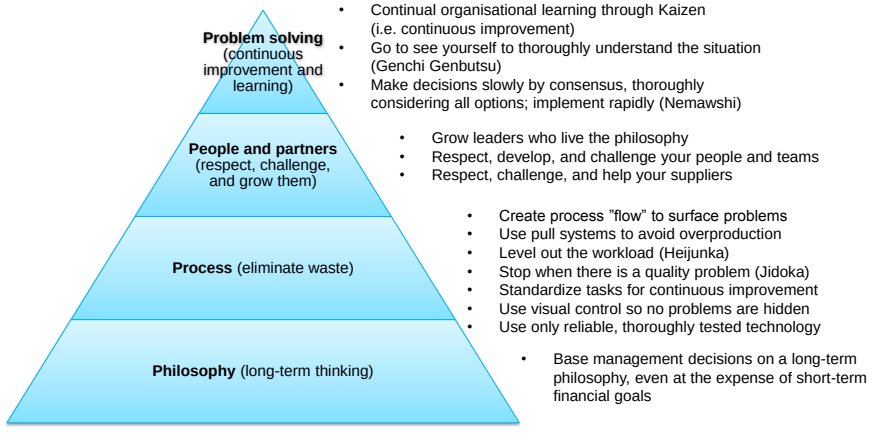


Figure 2. Principles of lean, modified from Liker (2004).

2.1.3 Planning in an agile–lean organisation

Agile and lean at the enterprise level requires looking at an organisation's entire value stream (Shalloway et al. 2009). Accordingly, the value stream refers to the stream of delivered software solutions from idea to implementation or from concept to consumption. Agile and lean at the enterprise level or business agility are referred to as an agile–lean organisation later in this thesis, since they convey and describe the same issue at hand. The enterprise in an agile–lean context means that all the parts of the organisation are involved in the value stream of a product and/or service that is created, enhanced, or maintained. Therefore, the intention is to minimize the cycle time from deployment to use and to remove waste and delays in the flow. Thus, the purpose is to create a system that helps minimize work-in-progress and maximize the speed at which business value is created (Shalloway et al. 2009).

The organisational planning consists of the required levels of planning and time frames. These are the most important aspects of planning that a company developing and improving its planning practices should define from the start (Lehtola et al. 2007). The planning level refers to items that are planned and the time frame refers to the length of the time period of the plans. However, there is no simple answer to how many different levels of abstraction for planning that a company should have, since both company size and its organisational structures play a role (Lehtola et al. 2007). Next, the levels of planning in agile–lean organisations are shown based on the related literature.

Cohn (2006) presents an agile approach to planning called the 'Planning onion'. The planning onion describes the hierarchical relationships between the different objects of planning. The planning onion consists of the following levels or layers of planning: strategy, portfolio, product, release, iteration, and day; as shown in Figure 3.

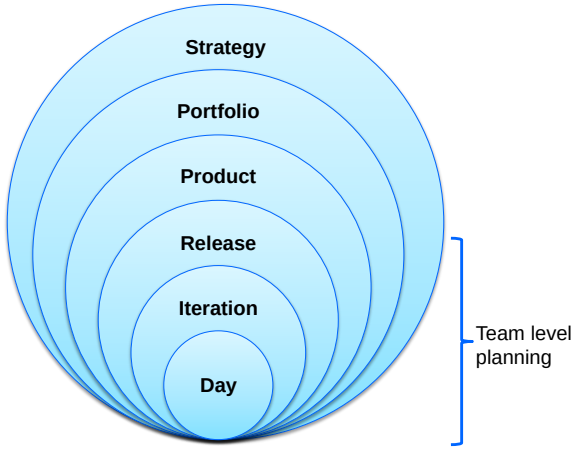


Figure 3. Planning onion, modified from (Cohn 2006).

According to Cohn (2006), planning should not extend beyond the planner's horizon – instead, it should include time for the planner to raise their head, look at the new horizon, and make adjustments. Thus, a progressive elaboration of the plan is needed. Agile teams achieve this by planning on three distinct horizons: the *release, the iteration, and the current day*. Release planning considers user stories or themes for a new release with the goal of determining the scope, schedule, and resources for a project. The release plan should be updated throughout the project so that it will always reflect the current expectations about what will be included in the release. During the iteration planning, at the start of each iteration, the product owner identifies the work that the team should address in the new iteration. The daily planning meeting is for organising work and synchronising daily efforts. Cohn (2006) explains that *product, portfolio, and*

strategic planning are outside the concern of most agile teams. Product planning involves looking further ahead than the immediate release and planning for the evolution of the released product or system. Portfolio planning involves the selection of the products that will best implement a vision established through a company's strategy.

Leffingwell (2011) presents the 'Agile enterprise big picture', which is both the organisational and process model for managing software requirements in an agile and lean manner. It consists of three levels: the portfolio, the programme, and the team, as illustrated in Figure 4.

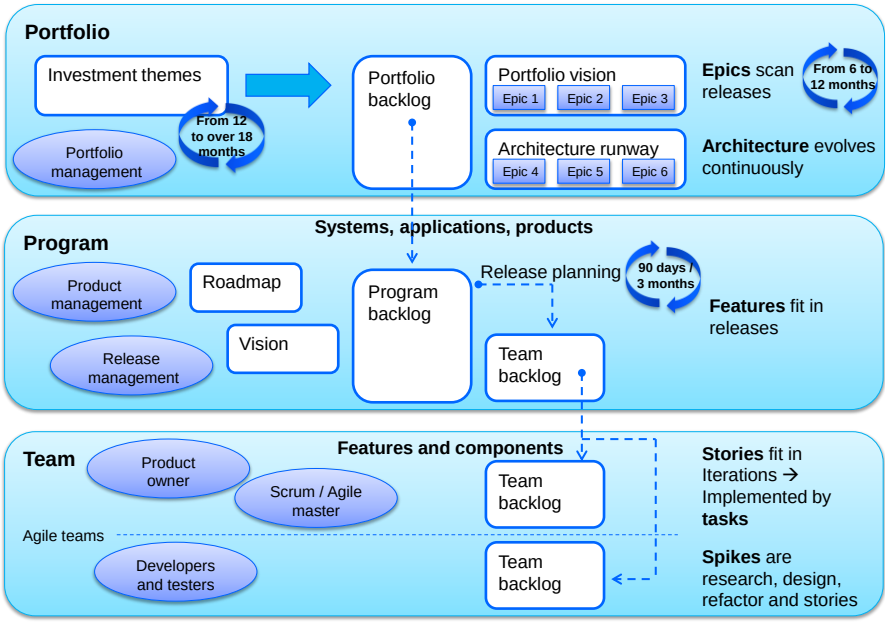


Figure 4. The agile enterprise big picture, modified from Leffingwell (2011).

According to Leffingwell (2011), *at the team level*, agile teams define, build, and test user stories in a series of iterations and releases. The team's product owner has responsibility for managing the backlog of user stories and other things the team needs to do. *At the program level*, the larger-scale system's functionality is developed by multiple teams in a synchronised agile release train (ART). The ART includes time-boxed iterations and milestones that are date- and quality fixed but where the scope is variable. The ART produces releases or potentially shippable increments (PSIs) in fixed 60- to 120-day time boundaries. The PSI defines the basic iterative and incremental cadence and delivery mechanism for the programme. Product managers are responsible for defining the features of the system at this level. *At the portfolio level*, a mix of investment themes is used to drive the investment priorities for the enterprise. The purpose is to ensure that

work being performed is necessary to the enterprise and according to its business strategy. Investment themes drive the portfolio vision and are expressed in a series of larger scale initiatives called epics. Epics express the product initiative in bullet form, as a sentence or two, in a video, as a prototype, in a short business case, for instance. The epics are allocated to various release trains over time. The portfolio managers are responsible for deriving the decisions. The decision process results in a set of themes or key product value propositions that provide market place differentiation and competitive advantage. The epics express the highest-level customer need. The epics are intended to deliver the value of the investment theme and are identified, prioritised, estimated, and maintained in the product backlog. The epics express the intent of the product initiative in any suitable form (e.g. in bullet form, in user-voice story form, with sentences, in a video, in prototype) (Leffingwell 2011).

Furthermore, Leffingwell (2007, 2011) points out that, from the planning point of view, agile affects among others planning and scheduling in software development and in software project management. *At the team level*, a series of iterations is used to combine larger, system-wide functionality for release to the external users. Many teams have four to five development iterations, and the cadence of a potentially shippable increment is about every ninety days, which creates a quarterly planning rhythm for the entire company. The responsibility for planning is transferred to the teams, and *at the programme level* it is accomplished by the release planning function. Release planning is done on a standard cadence, independent of the project status and release commitments. *At the portfolio level*, investment themes span to the strategic planning horizon, and the time frame is from twelve months to over eighteen months. Investment themes derive epics that express the highest level of expression of the customer or business need. The epics are planned in time frames of six to twelve months.

Shalloway et al. (2009) present an agile–lean organisation that involves coordinating business, management and delivery teams in order to deliver a sustainable stream of products based on prioritised business need and lean principles. The business area includes activities such as continuously prioritising and decomposing incremental needs across the organisation, managing a portfolio of business needs, and doing release planning. The product management area addresses activities including: organising cross-functional teams delivering incremental end-to-end features, managing the value stream, and bringing visibility to impediments. The delivery team's activities include: working together every day and delivering fully tested and integrated code, learning how to deliver business needs incrementally, and becoming proficient at acceptance test driven development and refactoring. A software development timeline in an agile–lean organisation together with the main levels of planning according to (Shalloway et al. 2009) is shown in Figure 5.

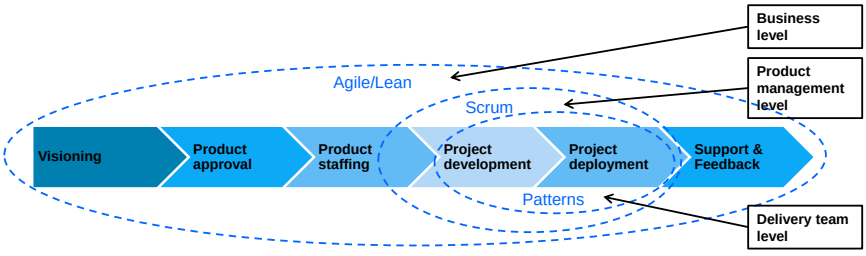


Figure 5. Software development timeline, adopted from (Shalloway et al. 2009).

As a summary of the models presented previously, in Leffingwell's (2011) planning model, the responsibility for planning is moved to the teams. According to Leffingwell (2007), planning mainly appears at two levels: gross-level planning for releases (the programme level) and fine-grained plans for iterations (the team level). At the programme level, planning is accomplished by the release planning function. Release planning is done on a standard cadence, independent of the project status and release commitments. Similarly, according to Cohn's (2006) planning onion, agile teams achieve the planning at three distinct horizons: release, iteration and day. The other levels of planning (that is product, portfolio and strategic planning) are outside the concern of most agile teams and are hence not defined in more detail in the model either. However, in terms of daily and operational planning, Ruhe (2010) has pointed out that they can both become difficult without a proper release plan that is well-aligned with the product and portfolio strategy of a company. Therefore, Shalloway et al. (2009) state that the team agility or team level planning is necessary but not sufficient, hence the real goal should be to create enterprise agility.

Shalloway et al. (2009) state that agile at the enterprise level requires looking at an organisation's entire value stream. They define a business level, product management level and delivery team level of planning pertaining to agile and lean organisations, but in the planning practices, they focus on portfolio, product and release planning. Shalloway et al. (2009) present a continuous planning process pertaining to releases (defined in more detail later in Subchapter 2.3.5), which is performed before each iteration and during daily stand-up meetings. Accordingly, continuous planning, at the project level, is for what is known (for example, from two to four weeks forward), and the work is for the next iteration and for today. Shalloway et al. (2009) do not say how, for example, strategic, financial, or business planning should be conducted. Similarly, Cohn's (2006) planning onion and Leffingwell's (2011) agile enterprise big picture are missing how those levels of planning should be conducted in an agile–lean organisation. Instead, Heikkilä et al. (2013) have adopted a three-level planning model, including strategic planning, release planning and operational planning for a large-scale agile software development organisation. Accordingly, strategic planning involves interaction between business and management, and development and is performed in the long term. Release planning refers to the feature content of the next release and

to planning aiming to create content efficiently. Operational planning concerns the implementation of features on a day-to-day basis. However, as with the other models shown, Heikkilä et al. (2013) focus on release planning without going into detail on strategic or financial planning. Thus, their understanding of organisational planning also lacks a broader perspective.

Based on the current models presented in the related literature, the levels of organisational planning are the following: strategic planning, business and financial planning, portfolio planning, product planning, and release planning. These levels of planning are defined in more detail from the continuous planning point of view in Subchapter 2.3.

2.2 Roadmapping

A roadmap is a layout of existing routes or paths, which is used to decide among alternative directions towards a desired destination (Kostoff and Schaller 2001). In more detail, a roadmap is a visualization of a forecast, which can address a number of key areas, such as technology, capability, platform, system, environment, threat and business opportunity (DeGregorio 2000). According to Kappel (2001), roadmaps are also forecasts of what is possible or likely to happen, and plans that express a course of action. According to Phaal and Muller (2009), the roadmap provides a strategic lens through with complex systems, such as a business or strategic issue, can be viewed. It helps to structure and represent multiple interrelated perspectives on the evolution of the system, providing a framework to support understanding and communication (Phaal and Muller 2009). Furthermore, roadmaps are intended to be living documents, to be reviewed and updated over time in order to remain useful (Albright 2003).

Roadmapping describes the process of creating and revising roadmaps (Kostoff and Schaller 2001). It is a strategic planning and forecasting process with long-lasting future activities (Kappel 2001). Roadmapping is also claimed (Li and Kameoka 2003) to be a decision-making and design process. Furthermore, Strauss et al. (1998) describe roadmapping as a management activity that links customer/market needs and opportunities, product quality and competitive positioning as well as corporate capabilities such as business and technology value chain attributes. Forecasting can be considered to relate to technology or market trends, and planning, on the other hand, to products, product lines, resources or the entire company (Van de Weerd et al. 2010). Roadmapping supports strategic dialogue within an organisation, and particularly across functions and levels, but also between organisations. Phaal et al. (2008) highlight that, even though roadmapping can be used as a stand-alone problem-solving method, it will have more impact if it is integrated with core business processes where decisions are made and budgets allocated. In particular, relevance should be given to strategic planning in which decisions are routinely taken that have a significant long-term impact, and where the situation is complex and the future uncertain (Phaal et al. 2008).

2.2.1 Roadmap structure

Roadmaps can be expressed in various forms, types or with different taxonomies (Kameoka et al. 2003). Even though the roadmaps may take various forms or taxonomies, they should all answer a common set of 'why-what-how-when' questions (Phaal et al. 2005) that generally relate to markets, products, and technologies. However, the form of the roadmap should be tailored to the specific needs of the company and its business context (Phaal et al. 2004b). Luckily, the form of the roadmap is flexible, and the structure of the roadmap and the process used to develop it can be adapted to many different contexts (Phaal et al. 2004a)

The roadmap structure enables scalability and allows the approach to be customised so as to suit the particular focus of interest. The roadmap structure consists of two dimensions: *layers* and *time frames* (Phaal and Muller 2009). In the roadmap structure, each layer of information provides input to the next level (Cosner et al. 2007). The structure of a multi-layered roadmap is shown in Figure 6.

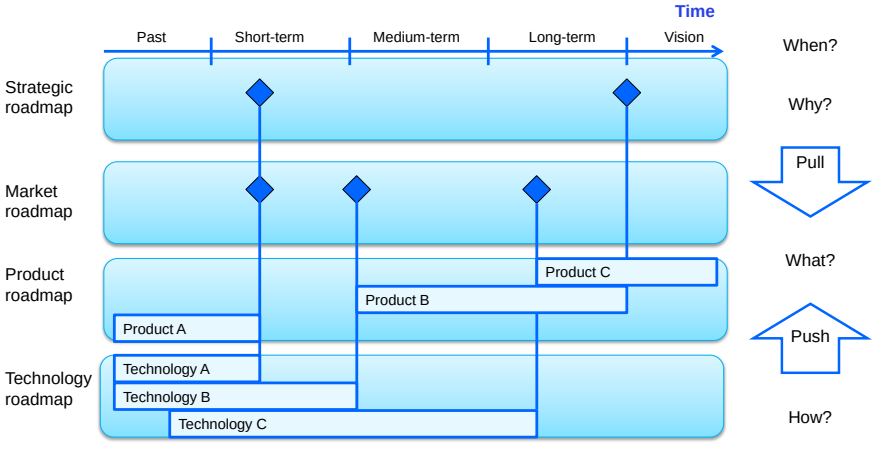


Figure 6. Multi-layered enterprise roadmap, adapted from (Cosner et al. 2007, Phaal et al. 2008, Phaal and Muller 2009).

The layers according to Kappel (2001) are shown in a chart, in which the horizontal axis describes the roadmapping purpose at the industry or company level, and the vertical axis describes the content emphasis of the roadmap either on specific trends or on positioning within an industry. According to Phaal et al. (2005), roadmaps commonly take the form of a multi-layered time-based chart that includes different layers of knowledge relating to purposes, deliveries, and resources. More precisely, Phaal and Muller (2009) state that at the highest level roadmaps comprise three broad layers: the top layer (external market and industry trends and drivers), the middle layer (products, services, infrastructure or other mechanisms for integrating technology capabilities, knowledge and resources),

and the bottom layer (both knowledge-based resources: technology, skills, competencies, and other resources: finance, partnerships, and facilities).

More precisely, the enterprise roadmap development according to Cosner et al. (2007) includes the following layers of information: strategic roadmap, market roadmap, product roadmap, technology roadmap, and enterprise roadmap. The *strategic roadmap* describes the long-term objectives of senior management. The *market roadmap* shows known and predicted customer needs together with, for example, competitive strategies, the regulatory environment, complementary product evolution, substitute products, and innovations. Strategic goals and market targets are defined as milestones or target dates for certain events. *Product roadmaps* are for documenting performance and feature evolution and presenting new-to-the-company products and new-to-the-world products. *Technology roadmaps* include expected research and development (R&D) products, their availability dates, driving factors for R&D, and related information. *The enterprise roadmap* combines these different types of roadmaps across the enterprise. While each roadmap presents existing plans, the enterprise roadmap may suggest alternate unfunded plans that would be considered if the enterprise were to alter its plans.

The multi-layered roadmap can be constructed from a market-pull or from a technology-push perspective (e.g. Albright 2003, Groenveld 2007, Phaal et al. 2008). From the market-pull point of view, a roadmap should begin by defining the most important requirements of the marketplace and customers. This strategy includes defining product development in the process of time and defining the required technologies for these products. From the technology-push viewpoint, a roadmap should begin by defining the key or new technologies and their market needs. This strategy describes how technology is going to affect the functionality of the product (Albright and Kappel 2003, Groenveld 2007).

Phaal and Muller (2009) recommend including five broad *time frames* in roadmaps: the past (and current situation), short-term, medium-term, long-term, and vision. Accordingly, the past time frame helps to understand the key influences and events that have led to the current situation, and highlights learning points that will influence the success of future plans. Short-term is typically a one-year horizon. It is the most important output of the roadmap, since it includes tangible plans and committed actions. This time frame can be called the budget horizon, since resources need to be committed so that the actions can be fulfilled. Medium-term is typically around three years, with links to the strategic planning horizon. It highlights the broader direction and options that influence the short-term directions and plans. The long-term time frame is typically a ten-year time frame. It provides a bridge between the medium-term strategy and the vision of the organisation. It enables the organisation to articulate key uncertainties and scenarios, and to explore long-term shifts in the technology, business and market environment, as well as to capture and assess longer-term issues affecting current decisions and plans. Vision is about knowing where you are going, setting out the long-term aspiration of the organisation (Phaal and Muller 2009).

2.2.2 Roadmapping process

The roadmapping process focuses on a sharing of perspectives, involving interaction between people, leading to communicate, acquiring new understanding, insights, creativity and learning (Phaal et al. 2005). However, the roadmapping process is different from one company to another (Phaal et al. 2004b, Groenveld 2007). According to Groenveld (2007), this is because companies serve different markets and have different cultures. Furthermore, according to Phaal et al. (2004b), the most suitable roadmapping process for a company depends on many factors. For instance, it depends on the level of available resources, such as people and time, on the issues to be addressed, such as purpose and scope, on available information, such as market and technology, and on other relevant processes and management methods, such as new product development, project management and market research (Phaal et al. 2004b).

Albright and Kappel (2003) define the roadmapping process in a concise way. The process includes initiation, maintenance and restarts if required. Similarly, Phaal et al. (2003) identify three roadmapping phases: planning, facilitated roadmapping workshop(s) and roll-out. Alternately, McCarthy (2003) adds two more phases to the roadmapping process that consists of team formation, focus, technology or workflow analysis, implementation and review.

In particular, from a software development perspective, roadmapping typically relates to products, and the roadmapping process is defined as follows in the related literature. Lehtola et al. (2005) describe a three-phase roadmapping process including phases such as preparation, approval and communication. In contrast, van de Weerd et al. (2010) define the roadmapping process as consisting of theme identification, core asset identification, and roadmap construction. In addition, Vähäniitty et al. (2002) propose a four-step model especially for creating and updating product roadmaps. These steps are as follows: 1) Define the strategic mission and vision of the company, and outline the product vision, 2) Scan the environment by identifying major trends, 3) Revise and distil the product vision as product roadmaps, and 4) Estimate the product life cycle and evaluate the mix of development efforts planned. The steps in the model should be performed periodically in order to adjust the roadmap to new information and changing market situations. Smaller updates to the roadmaps are suggested to ensure up-to-date information (Vähäniitty et al. 2002).However, unlike others, Lehtola et al. (2005) keep the release roadmapping process as distinct from product roadmapping. It is proposed to consist of data collection, feature prioritisation, release planning and release roadmap validation. Similarly, van de Weerd et al. (2010) keep the requirements management and release planning as distinct from product roadmapping. The requirements management consists of gathering, identifying, and organizing requirements. The release planning includes requirements prioritization, requirements selection, release definition, release validation, launch preparation and scope change management.

According to Corner et al. (2007), there are three different approaches to creating roadmaps: 1) a central process (built by a central group, using information collected across the enterprise), 2) a workshop approach (built in a series of workshops with different groups of stakeholders), and 3) a distributed approach (built by each contributing organisation, working to enterprise guidelines). These approaches are defined according to (Cosner et al. 2007) as follows:

In the central process, the team builds all the roadmaps using information provided by the business unit content owners. The team meets with the stakeholders to obtain and understand the planning data. Then, the team creates the roadmaps and distributes them to the business units for information. So, the roadmaps are produced with consistent content and format, and they are used for creating summary roadmaps that cut across the different business units. With the help of this process synergies between the roadmaps are easier to find.

In the workshop approach, roadmaps are produced in collaborative sessions with the business-unit content owners, and it is facilitated by the enterprise team. The enterprise team conducts a series of workshops in each business unit. In the beginning, the workshop is tutorial in nature, and focuses on the roadmap's purpose, structure and content. Afterwards, the central team and the business unit participants work together to produce the business unit's roadmaps, using the information that the participants brought to the meeting.

In the distributed approach, the individual business-unit and functional content owners create the roadmaps in a way that subsequent integration is enabled based on guidelines provided by the enterprise team. A small number of people from each participating organisation own and manage the process. They are also responsible for training and supporting the participants in the business units, who create and maintain the roadmaps. The approach promotes a sense of ownership of the roadmaps by the business units, but it also leads to major differences in roadmap content, format, and quality.

2.2.3 Roadmapping participants

The roadmapping process gathers together stakeholders from different functions of the organisation or from different organisations (e.g. Li and Kameoka 2003, Lehtola et al. 2005). Thereafter, *a roadmapping team* is formed from the gathered stakeholders. The team shares information and perspectives to make decisions that are then presented in a roadmap (Lehtola et al. 2005). According to McCarthy (2003), only the roadmapping team participates in the roadmapping process, but support from management is needed regarding personnel and budget investments. The team should be formed at the beginning of the roadmapping process, including R&D and technology management personnel, members from business development, representatives of finance, and core staff members from other functions. The first task of the team is to establish a common understanding of the process and the terminology to be used. After that, the team should begin to develop a detailed analysis of the process, and to decide on the factors and

metrics required for the process evaluation. The roadmapping team is also responsible for analysing the required technologies as well as implementing and reviewing the roadmaps (McCarthy 2003).

Similarly, Groenveld (2007) proposes that the roadmapping process should be started with a small roadmapping team, in which marketing, product management, research, development, and engineering teams participate. But later, the team looks for a leader who should become *the owner* of the drafted roadmaps. The owner is responsible for the maintenance of the roadmaps and for the initiation of appropriate updating actions as well as for providing additional information when needed. Accordingly, the roadmapping team guides the process and organises workshops to ensure integral involvement of the organisation and input by the organisation. The outcome of the workshops is used to prepare draft roadmaps, or parts of them (Groenveld 2007). According Phaal et al. (2000, 2003), a multifunctional team is needed in the roadmapping process in order to provide multiple perspectives, such as commercial, technical, research, development, manufacturing, marketing, and finance. In addition, Phaal et al. (2003) believe that both the business owner and the process owner should participate in the roadmapping process. The owners should be involved in the planning phase and, thereafter, throughout the roadmapping process. It is proposed that the business owner is responsible for the business outcome of the process, while the process owner is responsible for the implementation of the roadmapping (Phaal et al. 2003, Wells et al. 2004). The owners are also responsible for selecting the members of the roadmapping team, for solving issues regarding the application, and for having knowledge of the roadmapping domain (Phaal et al. 2003).

In addition*, a facilitator* is proposed for managing and facilitating the roadmapping process (Phaal et al. 2003, Wells et al. 2004). During the different phases of the roadmapping, a facilitator can support and guide the roadmapping team (Albright 2002). A facilitator plays an active role in appropriately scoping the roadmap, forming the team, setting up a work plan, and assessing individuals with their tasks in the larger effort (Albright and Kappel 2003). Accordingly, the facilitator should also challenge assumptions and enforce rigour in the roadmap. According to Albright (2002), the roadmapping process is best performed as a cross-functional team led by an experienced facilitator. Through the roadmapping process, the facilitator steers the team towards a realistic plan. The cross-functional team includes many functions that contribute to the success of a product line or business: central and regional marketing, product management, R&D, manufacturing, services, etc. The purpose of the roadmapping team is to lay out a possible future or multiple futures, set objectives, and define a plan to achieve the objectives, as well as make sure that the required capabilities and technologies are available at the right times (Albright 2002).

On the other hand, Jantunen and Smolander (2006) have identified three types of roles that need to be present in a roadmapping process: contributor, controller, and distributor. The contributor is the person who brings valuable information to a roadmapping context. The controller ensures that roadmapping is done systematically, and the distributor absorbs information in a roadmapping context

and disseminates the results of roadmapping to those who need to act upon it. The contributor has similar tasks as the previously mentioned member of a roadmapping team, and the controller acts in a quite similar role to the facilitator. In addition, Van de Weerd et al. (2006, 2010) have differentiated between internal and external stakeholder groups. The internal stakeholders are described as follows: product management, company board, research and innovation, services, development, support, and sales and marketing. The external stakeholders are defined as follows: market (that is potential customers, competitors, and analysts), partners (that is implementation, development, and distribution partners), and customers.

2.3 Continuous planning

Planning in general is seen to consist of two things: actions and forecasts (that is expected outcomes). Forecasting can be considered to relate to technology or market trends, and planning to products, product lines, resources or the entire company (Van de Weerd et al. 2010). Continuous planning is about implementing planning practices continuously, not just as part of a top-down annual event (Hope and Fraser 2003). Planning should be carried out continuously so that, at any given time, the full scale of the development can be presented (Westkamper and Von Briel 2001). Fitzgerald and Stol (2014) define continuous planning as a holistic attempt involving multiple stakeholders from business and software functions, whereby plans are dynamic open-ended artefacts that evolve in response to changes in the business environment, and thus involve a tighter integration between planning and execution. Furthermore, in software development continuous planning refers to the organisational capability to conduct planning in rapid cycles, which can be hours, days, or a few weeks or months depending on the level of planning. According to Suomalainen et al. (2015), continuous planning in a lean software development context refers to a planning process that is dynamic and links strategy with execution through continuous iterative cycles.

Already several years ago, it was claimed by Myers (1999) that continuous planning is needed and that it would be increasingly important in the future. The background was that continuous operations required an ability to produce open-ended plans that develop and evolve within the dynamics of the environment. Thus, incremental planning techniques were also needed in continuous planning in response to changing situations. Myers' (1999) Continuous Planning and Execution Framework (CPEF) was developed so as to combine plan-generation and plan-use capabilities so as to solve complex tasks in unpredictable and dynamic environments. Based on CPEF, continuous planning was seen as consisting of the two following aspects: firstly, plans should be seen as dynamic, open-ended artefacts that evolve in response to an ever-changing environment and that they remain viable and relevant. Secondly, users are seen as an important part of the overall process, for example providing inputs that will

influence the types of plan that is generated, the number of options to consider, failure assessments and plan-repair strategies.

Later on, after adopting agile and lean development practices, planning has evolved towards constant planning in small increments and with more people than with traditional software development methods (Shalloway et al. 2009). Also, one reason why companies have adopted agile and lean practices is that annual planning cycles result in longer product development projects than necessary (Shalloway et al. 2009). According to Shalloway et al. (2009), a typical annual planning cycle runs from July 1 to December 31, which means that the last six months of the year is spent in collecting ideas and developing plans for work that begins next year. Thereafter, all the ideas that came along need to wait until the next planning cycle, next July 1. So, if planning is done annually, it means that the entire time from idea to delivery takes an average of eighteen months. Instead, if the planning is done quarterly, the average time the issues need to wait is only one and a half months. Also, other industrial experiences of software development companies (e.g. Lehtola et al. 2007, 2009) have shown that companies perform so called open-ended planning, with a pre-defined rhythm. Then, planning is undertaken at regular intervals, but the planning horizon for the future is not fixed. The planning is usually performed from one to two releases ahead. The near future is planned in more detail and the remote future is outlined, but in less detail. Accordingly, open-ended planning is well suited to market-driven planning where the decisions are trade-offs between now and later (Lehtola et al. 2007, 2009). But still, in the software development context, planning is commonly episodic and performed according to a traditional cycle usually triggered by annual financial year-end considerations (Fitzgerald and Stol 2014). In fact, the problem related to planning is that time is divided into a number of planning horizons, each lasting a significant period of time (Fitzgerald and Stol 2014), and the continuity is not seen throughout the organisation.

Today, as the transformation towards continuous practices is emphasized in software development organisations, it is highlighted that organisations should establish a continuous flow of software development (Fitzgerald and Stol 2014). Accordingly, it should be leveraged as an end-to-end concept that also considers other functions within an organisation, such as planning, deployment, maintenance, and operation. Furthermore, Fitzgerald and Stol (2014) claim that, rather than focusing on agile methods as such, a useful concept for assessing continuity in software development comes from a lean approach called flow. They clarify that, rather than a sequence of discrete actions performed by clearly distinct teams or departments, the argument for continuous software development is to establish a continuous movement, which closely resembles the concept of flow. Flow is defined as a progressive achievement of tasks along the value stream so that a product proceeds from design to end up in the hands of the customer without stoppages, scrap or backflows (Womack and Jones 2003). However, Fitzgerald and Stol (2014) point out that many traditional software development environments are still operating according to the principles of batch-and-queue. Despite the fact that the software development would be flowing to some degree,

the planning and deployment of features are still dealt with in batches, and not in a continuous flowing movement (Fitzgerald and Stol 2014). Therefore, drawing on the lean concept of flow, Fitzgerald and Stol (2014, 2015) identify an umbrella term called Continuous * (continuous star) for a number of initiatives that are termed continuous in the context of software development, continuous planning being one of these initiatives. However, they do not define in detail to which level of planning continuous planning relates and how continuous planning should be conducted or what the mechanisms are for continuous planning, for instance.

Next, the levels of organisational planning in agile–lean organisation, identified in Subchapter 2.1.3, are described in more detail. These levels of planning are the following: strategic planning, business and financial planning, portfolio planning, product planning, and release planning. In the forthcoming subchapters, these levels of planning are discussed especially from the point of continuous planning and the agile–lean organisation.

2.3.1 Strategic planning

A generic strategy process can be divided into four stages: analysis, development, planning and implementation (Eppler and Platts 2009). Bryson (2011) defines strategic planning as follows: ‘a deliberative, disciplined approach to producing fundamental decisions and actions that shape and guide what an organisation (or other entity) is, what it does, and why.’ Strategic planning means accounting for where you are, where you want to be, how to get there and how these are connected (Bryson 2011). It also includes the development of timelines, resource allocations, responsibilities and deliverables (Eppler and Platts 2009). Mintzberg (1994) claims that the most successful strategies are visions, not plans. Therefore, strategic planning does not mean necessarily strategic thinking. After understanding the difference between planning and strategic thinking, the company can get back to the strategy-making process: capturing what the manager learns from all sources. According to Mintzberg (1994), planning is about analysis and breaking down a goal or set of intentions into steps, formalizing those steps to be implemented almost automatically, and then articulating the anticipated consequences or results of each step. Strategic planning, on the one hand, can be applied to all kinds of activities. For example, conventional planners organise planning and seeing how quickly it becomes formalised. Strategic thinking, on the other hand, is about synthesis and involves intuition and creativity. Strategic thinking results in an integrated perspective within the company, but not too precisely an articulated vision of direction. Planning cannot generate strategies, but it can make strategies operational (Mintzberg 1994). According to Beeton et al. (2008) planning or, as they call it, roadmapping strategies can be divided into two categories. Either the roadmapping can be exploratory, that is, surveying future possibilities, or goal-oriented, that is defining strategies to realise clearly defined future targets.

The strategy process varies across companies, but at a group level, it is commonly a continuous and issue-driven process (Bogsnes 2008). The various components of a given strategy commonly consist of specific routines and work patterns that vary from firm to firm and between different types of firms (Nordqvist and Melin 2010). Te Brömmelstroet (2013) states that strategic planning phases can vary widely in terms of how they are organised: bottom-up or top-down. However, all strategic planning processes can be seen as part of a multilevel group process, in which planning actors work together towards a shared outcome. Furthermore, Bogsnes (2008) states that, during the strategy process, the strategic objectives are often defined in what is known as a strategy map. Strategic themes are then commonly addressed as needed, or bi-annual executive committee strategy sessions will be held. When there is a major change in an organisation's strategic direction, its strategic objectives are often renewed or revised. Hence, the strategic plans should be living documents adjusting to internal and external changes (Nordqvist and Melin 2010). However, Eppler and Platts (2009) define the strategic planning process as one of the most demanding tasks that managers are facing because of today's complex markets. Thus, it is extremely challenging to take into account, simultaneously, the developments of technologies and societal trends, and the behaviour of competitors, customers and regulators, all within a changing legal, environmental and financial framework. Making sound and sustainable strategic decisions becomes even more challenging when this is compounded with time pressures, market uncertainty and constant distractions and internal tensions (Eppler and Platts 2009).

From an agile–lean organisation's perspective, Koenigsaecker (2009) defines a lean strategic organisational process in which strategic deployment is about reviewing key strategic efforts for the next year, identifying how lean improvement can be accelerated and enabling these efforts, setting true goals to support the strategic direction, establishing the pace and pattern of improvement effort for the year to achieve these goals, and then establishing a monthly review process. According to Koenigsaecker (2009), strategic planning is typically done once a year and is a learning experience by itself. Also, monthly strategy deployment meetings are held to review progress, to create opportunities, and to share knowledge about the lessons learned. The existence of monthly strategy-deployment reviews helps to get a company thinking about how to make its work fundamentally better with each passing month. Most companies commonly have a monthly meeting to review performance that is financially driven. Even though the meeting would be about a company's direction, it is fundamentally focused on modifications to the financial plan. Instead, a strategy deployment is a process that focuses the enterprise on fundamental improvement and on learning from the on-going improvement experience. By having the monthly strategy-deployment reviews, it helps to get the enterprise thinking about making the work fundamentally better every month.

2.3.2 Business and financial planning

Every business should have clearly defined objectives and parameters within which to operate. The business planning process provides an opportunity to assess the range of skills needed for a business to succeed and to identify potential gaps within this range. Financial planning and the preparation of marketing plans help to determine whether or not the objectives are achieved. According to Leffingwell (2011), in most enterprises investment decisions occur at the business unit level based on an annual or bi-annual budgeting process. During the budgeting process, the amount of funds available for each business unit or product to invest in development is determined (Leffingwell 2011). Financial planning is commonly conducted by the financial and business management.

Wareham and Majka (2003) claim that continuous financial planning processes are commonly based on goals formulated in a strategic plan, such as the establishment of capital structures appropriate to an organisation's current competitive and strategic position. They also point out that continuous financial planning should involve a capital allocation process that forces an organisation to prioritise capital spending decisions in a way that will improve the services provided while also protecting the long-term financial capacity (Wareham and Majka 2003). Furthermore, marketing plans provide measurable targets to compare and monitor progress as well as achievements on a continuous basis (Butler 2012). According to Cosner et al. (2007) the budgeting process should be aligned with the key milestones and events of an organisation's roadmap. More precisely, it should include the following three elements: firstly, the R&D categories for development resources for each major capability of a roadmap, secondly, categories of resources for operations that support different capabilities, and thirdly, life-cycle funding plans that forecast necessary 'ramp-ups' as well as simultaneous 'ramp-downs' in operational funding related to different types of capabilities shown on a roadmap.

In relation to financial planning, Rickards and Ritsert (2012) have discussed rolling forecasts and budgets. The most important characteristics of rolling plans compared to traditional forecast and budgets are as follows:

- a constant horizon independent of the financial year
- periodicity (the rule of quarterly preparation)
- planning is more detailed for early periods and less detailed for later periods
- planning focuses mainly on monetary and non-monetary business drivers that influence monetary results (revenues and costs).

According to Rickards and Ritsert (2012), rolling forecasts and plans (relating mainly to budgets) are commonly made over a time period typically spanning five quarters (or thirteen months) to eight quarters (or twenty-two months). Furthermore, they claim that many enterprises that utilise rolling forecasts and budgets use them in combination with a traditional budget plan. This approach determines using the lower boundary of five quarters (or thirteen months),

because at latest, at the beginning of the fourth quarter (or twelfth month), forecasts and budget values must completely cover the next financial year. Rolling revisions of plans ensure that the length of the time period covered is constantly evaluated so that new information is integrated into plans. By doing this, more detailed forecasts and budgets can be made for upcoming quarters, both in the near and distant future.

Hope and Fraser (2003) have presented the 'beyond budgeting' management model, in which they define budgeting in a much broader context than is commonly understood. One of the highlights of this model is that it understands budgeting, planning and improvement as a continuous and customised process, not just as part of an annual event (Hope and Fraser 2003). The model includes a set of coherent alternative processes supporting relative targets and rewards, continuous planning, resources on demand, dynamic cross-company coordination and an array of multilevel controls (Hope and Fraser 2003). The model is divided into twelve principles: six leadership principles (including customers, organisation, responsibility, autonomy, values and transparency) and six process principles (including goals, rewards, planning, controls, resources and coordination).

The beyond budgeting principles are quite similar to lean principles (presented in Subchapter 2.1) as they both advocate a customer-centric focus, systems thinking and self-organisation and empowerment of people within an organisation (Fitzgerald and Stol 2015). The beyond budgeting model emerged around the same time as the agile methods and both concepts have similarities, with both having an agile or adaptive perspective (Lohan et al. 2010, Lohan 2013). Lohan et al. (2010) have explored the performance management of agile software development (ASD) teams through the lenses of the beyond budgeting model; that is, they have taken the model as a whole and explored its applicability within the ASD domain. Lohan et al. (2010) have stated relating to planning that *'*Rather than having a single top-down fixed plan that determines actions for the year ahead, the devolution of the planning would allow for a continuous adaptation of short term plans to meet strategic objectives*.'* Agile development methods, for example, using Scrum and eXtreme Programming (XP), foster an environment for operational transparency through regular communication, improve accessibility to project information and increase the developers' continuing awareness of the work that is going on around them (Chong 2005). Thus, ASD teams should have access to all relevant information needed for them to operate efficiently, including also access to other ASD teams' velocity rates, burn down charts, product backlogs, etc. operating within the organisation (Lohan et al. 2010).

According to Bogsnes (2008), similar to the rolling forecasts and plans presented by Rickards and Ritser (2012), five-quarter rolling forecasts have become standard in 'beyond budgeting' implementations. However, Bogsnes (2008) clarifies that, even though a five-quarter rolling horizon is better than stopping planning at the year-end, planning is still done in a fixed period. Hence, a more dynamic planning process that is more event-based than calendar-driven with no fixed update frequency and with no fixed time horizons should be developed (Bogsnes 2008).

2.3.3 Portfolio planning

Portfolio management and planning is about selecting the most important products to create and enhance. Especially from the software product management perspective, according to Bekkers et al. (2010), portfolio management is strategic information-gathering and decision-making across the entire product portfolio, including the following focus areas: market analysis, product lifecycle management, and partnering and contracting. Market analysis gathers decision support information about the market needed to make decisions about the product portfolio of an organisation. Information-gathering and key decision-making about the product life and major product changes across the entire product portfolio are made in product lifecycle management. Partnerships, pricing, and distribution aspects are established through partnering and contracting (Bekkers et al. 2010). Furthermore, from an agile–lean organisation's perspective, Leffingwell (2011) has pointed out that a set of strategic investment themes drive the vision of all products, systems, and services and the responsibility of the investment decisions generally lies with a portfolio management team.

Portfolio management in an agile–lean organisation starts by thinking about the relationships between the portfolio's needs and the development team's needs (Shalloway et al. 2009). Accordingly, the goal is to support a fast-flexible-flow of work as well as to select projects that return the greatest value to the organisation. Lean guides the organisation to select smaller projects to work on whenever possible, which means that early in the planning, projects are defined as small as they can possibly be. Hence, essential projects are being worked on all the time. Agile–lean portfolio management brings value to the customers by prioritising which business features to work on, based on business value and then managing the project in a visible portfolio. This approach allows stakeholders and clients to identify and prioritise features that create the highest return on investment (ROI) for the business. In an agile–lean organisation, cross-functional teams can review and break down business features and system dependencies in order to create minimal marketable software solutions. The lean approach is a results-based, validation approach unlike the traditional task-based and resource-driven approach. Thus, status reviews are based on the validation of technical results, instead of completed tasks (Shalloway et al. 2009).

Poppendieck and Poppendieck (2009) suggest a novel approach to lean portfolio management to manage larger as well as more important projects and initiatives. This approach is recommended in Ward (2007). The first thing to do is to classify development efforts by type. The types may include, for example, business feature upgrades, strategic business initiatives, infrastructure upgrades and maintenance. The second step is to determine the cycle time (that is a time-box) for each type of effort. After each type of effort has been time-boxed, a schedule can be created that shows how many initiatives of each type you have the capacity to take in a year. For example, the classification of effort may be allocated as follows: two slots of six months each for strategic business initiatives,

twelve two-month-long slots for business feature upgrades a full year for the on-going upgrade effort and twenty percentage capacity for other efforts including maintenance. Based on this classification, one can see how much capacity is needed for the year for medium and large requests and initiatives as well as the staffing levels needed to provide that capacity. However, this approach does not really support the continuous planning perspective.

2.3.4 Product planning

Product planning is an important part of the software engineering (SE) process, as it is a method for planning and defining software product requirements based on the market and needs of stakeholders. Software systems requirements engineering (RE) is defined as a process of discovering a system's purpose, by identifying stakeholders and their needs, and documenting these in a form that is adequate for analysis, communication, and subsequent implementation (Nuseibeh and Easterbrook 2000). Product planning is also closely linked to RE, as it deals with the different releases of each product (Jantunen and Smolander 2006). In a product plan, the product is represented as product releases containing several product features. According to Wiegers (2003), a product feature is a set of logically-related requirements that provide a capability to the user and that enable the satisfaction of a business objective. On the other hand, a requirement is a statement of a customer need or objective. A requirement can also be a condition, which a product must meet to satisfy such a need or objective. In other words, a requirement is a property that a product must have to provide value to a stakeholder. However, product planning or product roadmapping differs from RE in that it is a process with long-lasting future activities, and in the product plan, high-level features are presented within a timeline and scheduled for different releases. In contrast, in RE, the features are analysed in more detail and defined as to what they mean from the perspective of the product development project. Lehtola et al. (2005) have also pointed out that targets for the release planning and the product planning are quite the same, but the product plan is more high-level. The main objective of the release plan is to inform stakeholders about scheduled future releases. On the other hand, the objective of the product plan is to help product managers create and maintain release plans, manage situations where the same technical product is included in several products, and use R&D to identify needs for research projects (Lehtola et al. 2005).

Product planning focuses on gathering information for a roadmap and creating a roadmap for a product or product line and its core assets (Bekkers et al. 2010). Therefore, a product roadmap is one of the main documents in product planning. According to Bekkers et al. (2010), product planning in the software product management context consists of three focus areas: roadmap intelligence, product roadmapping and core asset roadmapping. Accordingly, roadmap intelligence gathers the information needed for the decision support in order to create the product roadmap. Product roadmapping is the actual creation of the product

roadmap itself. Core asset roadmapping is about planning the development of core assets, for example, components that are shared by multiple products (Bekkers et al. 2010).

A product plan, also known as the product roadmap, provides a forecast of product family evolution over time and views the whole platform or relationships between the products in a platform (Albright and Kappel 2003). Typically, product roadmaps are developed, reviewed and improved iteratively. Often, this is achieved through close human interaction, such as face-to-face meetings and workshops with relevant stakeholders (Phaal et al. 2005). Product roadmaps are owned by the business owner of the product, who is also responsible for gathering all relevant stakeholders to obtain the needed information for the roadmaps. Product roadmaps typically cover a scope of two to three years, during which time they are frequently revised in order to ensure the currency of the documentation (Tabrizi and Walleigh 1997, Lehtola et al. 2005). Documentation of features is an important part of product planning, since accurate documentation ensures that the plans can be read, analysed, redrawn, and validated (Nuseibeh and Easterbrook 2000). According to Lehtola et al. (2005), the information in the product roadmap incrementally describes how the product and its business environment changes yearly. The fields in the product roadmap define the high-level functionality of the product and target customer group. The high-level product functionality is a description of forthcoming releases with basic mandatory information. The mandatory information includes the release goal and high-level features for each release. In addition, release time, localization, platforms, and dropped topics, e.g. features that are not supported in the subsequent versions of the product, are recorded. The fields include information about positioning, market arguments, and the geographical focus of the product for every year. This information is represented with a few bullet points for each issue (Lehtola et al. 2005).

From the software product management perspective, the product roadmap provides an overview of how a product is going to develop over the strategic time frame of up to five years in terms of new releases or versions, their schedules, and major themes (Kittlaus and Clough 2009). Accordingly, a roadmap usually shows dependencies between products or platform technology. In some companies, the roadmap may also include a rough financial picture, involving the expected revenues and costs. The roadmap tends to be rather detailed and precise for the short-term time frame, and the more it looks to the future, the less precise it tends to be. The roadmap gives directions to both internal and external use. Internally, the roadmap shows the relationship between product plans and financial forecasts and the major themes and the requirements governing the plan. Externally the roadmap demonstrates the viability of a product as well. The roadmap is important especially for a product manager to reach an agreement within the company regarding the longer-term direction and priorities (Kittlaus and Clough 2009). Commonly, the product roadmap is updated as part of the company's planning cycle (Kittlaus and Clough 2009), which in agile software development planning should be constant and in small increments (Shalloway et al. 2009).

Pichler (2010) describes product roadmapping from a more iterative, agile software development perspective. Accordingly, a product roadmap allows for capturing the goals of upcoming product versions. The product roadmap is a planning artefact showing how the product evolves across product versions and facilitating discussion between the Scrum team and stakeholders. A product roadmap helps organisations to coordinate the development and launch of products in the product roadmap or product portfolio. The product roadmap should be kept simple and focused on essentials. It should include the following information: the projected launch date for each version, the target customers and their needs, and the top three to five features. Product roadmaps are living documents since they evolve and change. The product roadmap should be created once the product has successfully been introduced into the marketplace. It should be created together with all the relevant people, for example, including the Scrum team, the person in charge of the product portfolio, representatives of other product development teams, and stakeholders. Furthermore, the roadmap should be created for a realistic time frame focusing on the next six to twelve months rather than predicting the next two to three years (Pichler 2010).

2.3.5 Release planning

Release planning is a process of planning for the next release of an evolving product (Ruhe 2010). According to Bekkers et al. (2010), release planning includes the software product management capabilities needed to successfully create and launch a release, which includes requirements prioritisation, release definition, release definition validation, scope change management, build validation, and launch operation. In requirements prioritisation, requirements are identified and organised. In release definition, the requirements to be implemented in the next release are selected based on the prioritisation, and the release definition is created. The development department performs the release definition validation before the release is built. The different kinds of scope changes that can occur during the development of a release are handled in scope change management. After realising the release, the development performs the build validation that focuses on validating the build release before it is launched. Internal and external stakeholders are prepared for the launch of a new release with launch preparation when issues from communication to documentation, training, and preparations for the implementation of the release are addressed (Bekkers et al. 2010).

Shalloway et al. (2009) present a continuous planning process pertaining to releases, especially for agile–lean organisations. Accordingly, release planning is a continuous and transparent activity that the whole organisation can observe. Anyone can contribute to the discussion about the value of items in the plan and the effort required in producing them. The agile–lean product portfolio serves as a transparent focal point for the business to sequence releases of minimum marketable features (MMF). During release planning, a product vision is

continuously decomposed while focusing on those features with greater priority (or value) to the business. Just-in-time methods are used to prevent wasted effort on lower-level or unneeded features, which means that the features are expanded just as much as is needed according to their build timetable. The release plan enables the team to look ahead so that activities requiring major effort can be broken down into smaller segments (right-sized work) and balanced against possibly upcoming higher priority features. A good release plan provides a clear visual control and removes the need to look too far ahead and work too far in advance of larger features. Figure 7 illustrates continuous release planning and the associated activities as presented by (Shalloway et al. 2009).

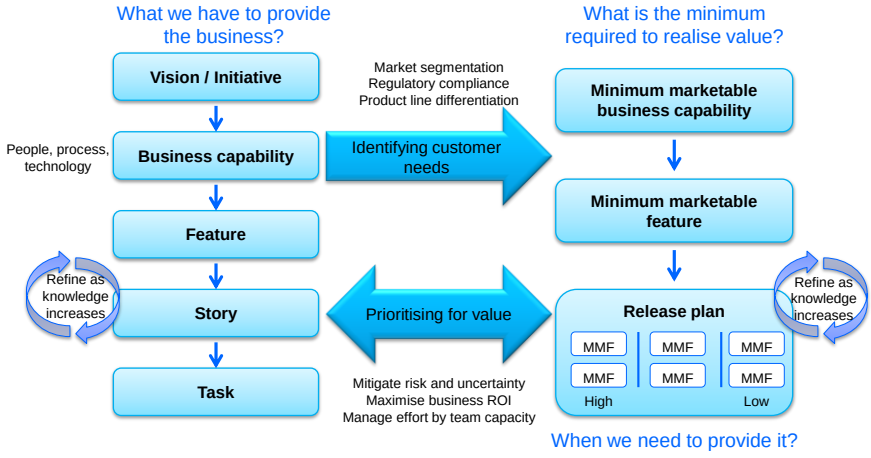


Figure 7. Continuous release planning, adapted from Shalloway et al. (2009).

According to Shalloway et al. (2009), continuous release planning starts with a vision provided by the product champion, who can make decisions regarding both the customer and the business value priority. The vision should be understood and reviewed by the delivery team and revised as market conditions change priorities. The vision should be visible (that is transparent so that everyone can access them) and reviewed as part of an iteration's planning session. Target dates can be determined by looking at the estimates in relation to the team's velocity. Short cycle times of one to four weeks enable quick feedback on the rate of completion and how well the customer needs are met. During each iteration, the team should focus on coding only the most important features at any time. It provides a clear picture of business value (that is features) evaluated against system constraints (that is technical stories) and enables high-value decisions regarding minimum releasable features.

According to Heikkilä et al. (2013), a continuous release planning process in a large agile–lean organisation is characterised by regular scoping and prioritisation of decisions and by incremental elaboration of features. Features are initiated

based on the availability of resources and the priority of the feature. The release planning is a collaborative action in which developers (that is the team) participate in the early phases of the feature elaboration. The work is divided among three work items: features, epics, and user stories. Epics are split from features and user stories are split from epics. Thus, according to Heikkilä (2010), plans for a release project can be seen on two levels or time horizons: roadmaps or PSIs. Roadmaps contain epics (i.e. epics form high-level functional goals for the products) and features, which show the tentative content of future releases. The PSIs are internal releases of the product, which contain a subset of the intended features of the final, external release of the product.

2.4 Summary of the related work

Agile and lean development practices are seen to complement each other; they share similarities but also differences. Agile software development focuses on the software development function, whereas lean development focuses on executing processes efficiently regarding the whole process from customer to delivery. Agility is about being able to create and respond rapidly to changes (Highsmith 2002a) and lean is about creating a change-tolerant organisation that can survive and succeed in times of uncertainty, change and complexity (Charette 2003). Despite the wide adoption of these practices, it is realised that the companies are going further with their practices towards continuous deployment. All of these practices, agile, lean and continuous deployment, change among other things scheduling and planning. Therefore, the main aim of this research is to study the change in planning, starting with roadmap-based planning and going towards continuous planning.

Roadmapping is a process of creating, reviewing and revising roadmaps. It is a planning and forecasting process with future activities. It is also a decision-making or design process, in which roadmaps are seen as either exploratory; surveying future possibilities, or goal-oriented; or defining strategies to realise future targets. Roadmaps can be illustrated with various forms or different taxonomies, but they commonly answer a set of ‘why-what-how-when’ questions that generally relate to markets, products, and technologies. In this thesis, the roadmap structure is presented with a multi-layered time-based chart that includes different layers of knowledge. The roadmapping process is different between companies, and thus there is not just one process model to be adopted. This is because companies serve different markets and have different cultures, but also because the process depends on the level of available resources, issues to be addressed, available information, and other relevant processes and management methods. Therefore, the roadmap process should be designed based on the company’s needs and the context in which it operates. Based on the related work, the roadmapping process commonly involves three phases. The first phase is the initiation of the process that involves activities such as planning and preparation. The second phase is the actual workshop(s), which involve the maintenance and the approval of the

roadmap. The third phase involves communication and implementation based on the roadmap decisions made in the workshop(s). Also, the third phase may involve a restart of the roadmapping process. The roadmapping process involves three main roles: owner, facilitator, and the member of the roadmapping team. The process is commonly conducted by a roadmapping team, which is formed by an owner and assisted by a facilitator. The roadmapping team is responsible for planning, creating, maintaining, and possibly redrawing the roadmaps. The owner is responsible for selecting the members of the roadmapping team and guiding the team through the workshops. In contrast, the facilitator is responsible for arranging practical matters, for example, materials and facilities for the roadmapping workshops, in order to enable the roadmapping process. However, in the case of a small roadmapping team, the facilitator might not be needed, and hence the owner, for instance, can conduct the facilitator's tasks.

Continuous planning is about implementing planning practices continuously, instead of a traditional annual top-down event, so that at any time the full scale of the development can be presented. Similarly with roadmapping, it involves multiple participants from business and software functions. But plans are seen as more dynamic open-ended artefacts that evolve in response to changes in the business environment. Thus, continuous planning involves a tighter integration between planning and execution. Furthermore, in software development, especially in the agile and lean context, continuous planning refers to the organisational capability to conduct planning in rapid cycles, which can be hours, days or a few weeks or months depending on the level of planning. The levels of planning (or layers of knowledge as referred to in the roadmap structure) are in the current literature divided based on strategic planning, business and financial planning, portfolio planning, product planning, and release planning (presented in Subchapter 2.3).

Based on the related work, often the responsibility for planning is moved to the agile teams, and thus planning mainly relates to iterations or releases. The other levels of planning, especially, portfolio, business, financial, and strategic planning, are outside the concern of most agile teams and hence are not defined with the same level of detail in the related literature either. Therefore, it is seen that the agile-team-level planning is necessary but not sufficient; hence, the real goal should be to create enterprise agility and to look at the organisation's entire value stream of planning. Based on the related work and the models presented in the current literature, it is evident that continuous planning requires a wider perspective than currently considered. It should be examined from a broader, even a more continuous perspective than currently exists. It is realised that continuous planning is not only a project- or team-level activity, but involves higher-level planning as well, for example, strategic and financial planning. Similar to the findings of this related work, Fitzgerald and Stol (2014) also state that the only forms of continuous planning have emerged from agile development approaches and are related to sprint iterations, or at best, software releases. Thus, it is evident that continuous planning is not widespread throughout organisations in the context of software development.

Figure 8 summarises the related work presented in Chapter 2. The figure brings out the current planning practices presented in the related work with the help of a roadmap structure. The horizontal axis presents the time frame of the plan, and the vertical axis presents the level of planning together with the main participants. Accordingly, the levels of planning involve the following participants: agile team, product management, portfolio management, business and financial management, and executive committee. The plans are created within the time frame, from days to eighteen months as illustrated in the figure. Release planning is commonly conducted within less than a three months' time frame, whereas product planning is conducted within a three- to six-month time frame. Portfolio, business and financial planning is done within a six-months to one-year time frame and strategic planning is done within the time frame of one year to eighteen months.

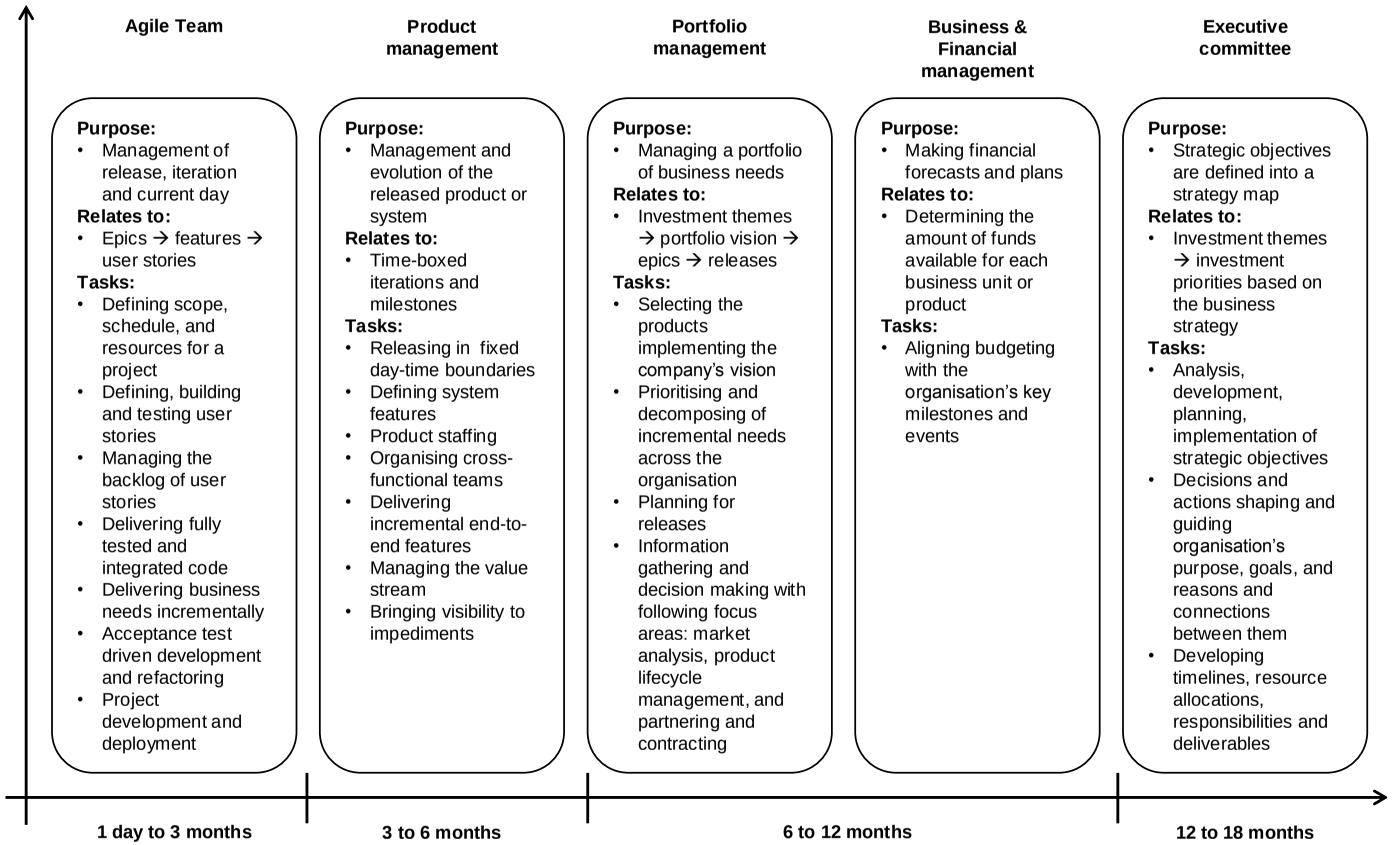


Figure 8. Summary of the literature findings.

3. Research design

Empirical research is characterised by an implicit or explicit research design; hence, this chapter introduces the research design of the study (Yin 1994). The research design specifies the logical sequence that connects the empirical data to the research questions of the study and, eventually, to its conclusions. Hence, the purpose of the research design is to avoid a situation in which the evidence does not address the initial research questions (Yin 1994). Therefore, the empirical research of this thesis was carefully planned and implemented. Firstly, the research approach is described, and then, the research methods used are defined. Thereafter, the case companies are presented together with the case selection criteria and description. Then, the research process together with data collection methods are presented, and finally the data analysis methods are defined.

3.1 Research approach

The research approach involves the major decision on how to conduct the research. According to Creswell (2003), there are three different research approaches: qualitative, quantitative and mixed methods. The qualitative data are descriptive, and capture as well as communicate experiences of the field of study. In other words, qualitative data tell a story about the researched phenomenon. In addition, qualitative research relies on logical conclusions on the gathered data. In contrast, quantitative data seek numerical responses, and thus rely on quantitative measurement and mathematical models (e.g. Yin 1994, Patton 2002). The mixed method research approach employs data collection and analytical techniques associated with both qualitative and quantitative research methods by using multiple data collection methods (Easterbrook et al. 2008).

The larger philosophical stances should be made explicit by the researchers since that information helps to explain the decision for the chosen research approach (Creswell 2003). Different people make different assumptions about scientific truth, and therefore it is vital to define how the assumptions are arrived at through scientific investigation (Easterbrook et al. 2008). In order to understand the different philosophical stances, it should be noticed that philosophers make a distinction between epistemology (the nature of human knowledge, and how we obtain it) and ontology (the nature of the word irrespective of our attempts to

understand it) (Walsham 1995, Easterbrook et al. 2008). Researchers also make claims about axiology (values affecting knowledge), and methodology (the process of studying knowledge) (Creswell 2003). There are four dominant philosophical stances adopted by scientists (Creswell 2003), as presented in Table 1.

Table 1. Dominant philosophical stances, based on (Creswell 2003).

Positivism	**Constructivism**
• Determination • Reductionism • Empirical observation and measurement • Theory verification	• Understanding • Multiple participant meanings • Social and historical construction • Theory generation
Advocacy/Participatory	**Pragmatism**
• Political • Empowerment issue-oriented • Collaborative • Change-oriented	• Consequences of actions • Problem-centred • Pluralistic • Real-world practice-oriented

The author adopts the constructivist philosophical stance. The stance that the author adopts affects which methods the author believes lead to acceptable evidence in response to the set research questions. Constructivism, also knowns as interpretivism (Yin 2003), rejects the idea that scientific knowledge can be separated from its human context (Easterbrook et al. 2008). Instead, constructive research highlights that software development is made and enacted by people with different values, expectations, and strategies, as a result of their different frames of interpretation (Vidgen and Wang 2009). The author concentrates less on verifying theories and more on understanding how different people make sense of the world and how they assign meaning to actions, which is typical of constructivists. Theories may emerge from the process, but they are tied to the context being studied (Easterbrook et al. 2008). Furthermore, the author prefers to use research methods that collect rich qualitative data about human activities from which local theories might emerge.

According to Creswell (2003), in a qualitative approach the researcher often makes knowledge claims based on constructivist perspectives (that is the multiple meanings of individual experiences, meanings that are socially or historically constructed, with the aim of developing a theory or pattern) or advocacy/participatory perspectives (that is political, issue oriented, collaborative, or change-oriented), or both. The researcher may also use strategies of inquiry such as narratives, phenomenologies, ethnographies, grounded theory studies, or case studies. The researcher collects open-ended, emerging data with the

intention of developing themes from the data (Creswell 2003). In contrast, in a quantitative approach, the researcher primarily uses post-positivist claims for developing knowledge (that is cause and effect thinking, reduction to specific variables and hypotheses and questions, use of measurement and observation, and the testing of theories), employs strategies of inquiry such as experiments and surveys, and collects data on predetermined instruments that yield statistical data (Creswell 2003).

In this thesis, the research is conducted as empirical research carried out as case study research. All four philosophical stances presented can be applied to case studies; however, different stances affect the way in which cases are selected and data analysis is performed (Easterbrook et al. 2008). Constructivists use exploratory case studies to investigate the differences in culture and perspective in various settings, as was done in this thesis. In contrast, confirmatory case studies draw on the positivist perspective of theory-driven research, though positivists also use exploratory case studies to develop new theories (Easterbrook et al. 2008). Thus, case studies can be based on both quantitative and qualitative evidence (Yin 1994, Creswell 2003). In case study research, data are collected through such methods as inquiries, interviews, observation, and through the use of documents and artefacts (e.g. Yin 1994, Patton 2002). In this thesis, both qualitative and quantitative research approaches are used, since the empirical data are collected with a questionnaire study, interviews, and through the use of company-specific internal memos and material. However, as the sampling of the questionnaire study is quite small, the research focuses mainly on the qualitative data.

3.2 Research methods

The qualitative research data in this thesis are collected by using a case study research method. A case study is, according to Yin (1994), 'an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between the phenomenon and context are not clearly evident'. Accordingly, a case study is both descriptive and interpretive by nature, and it allows an investigation to retain holistic and meaningful characteristics of real-time events such as individual events and organisational and managerial processes. According to Yin (1994), the descriptive need for case studies arises out of the desire to understand complex social phenomena. Furthermore, a case study is preferred when the researcher has little control over the events, and when 'how', 'why', and explanatory 'what' questions are posed (Yin 1994). Easterbrook et al. (2008) divide case studies into two categories: exploratory case studies and confirmatory case studies. Accordingly, exploratory case studies are used as initial investigations of some phenomena to drive new hypotheses and to build theories, and confirmatory case studies are used to test existing theories. However, according to Darke et al. (1998), case study research

is used to achieve various research aims, for instance to provide descriptions of the phenomena, to test a theory, and to develop a theory (Darke et al. 1998).

The case study research method was chosen for this thesis because the aim of the research was to provide descriptions of the phenomena by identifying general practices used by companies to conduct continuous planning and roadmapping. Also, one of the reasons for selecting a case study research approach was that the literature on continuous planning and roadmapping was not yet well formulated, and practical experience from the field of research was difficult to find. As Järvinen (2001) emphasises, case studies make it possible to examine very complicated circumstances and, in this way, to gather new information for creating new knowledge. Also, according to Yin (2003), the case design is eminently justifiable if the case represents (a) a critical test of existing theory, (b) a rare or unique circumstance, or (c) a representative or typical case, where the case serves a revelatory or longitudinal purpose. So, the use of case research was selected to provide experiences and to gather new information and gain an understanding of complicated circumstances, while it is emphasised that the environments in each organisation are unique depending on, for example, cultural, historical, and technical issues and backgrounds, production and strategic processes. Furthermore, according to Easterbrook et al. (2008), case studies are especially appropriate when the context is expected to play a role in the phenomena (for example, if the stresses of a real project affect developers' behaviour), or when effects are expected to be wide-ranging or are expected to take a long time (for example weeks, months, or years) to appear, as was the case in this research. Also, the case study methodology is claimed to be well suited to software engineering research, as it studies contemporary phenomena in their natural context (Runeson and Höst 2009).

Case study research can contain both single- and multiple-case studies. A single-case study is used when a well-formulated theory is tested, or when there is the possibility to have access to an extreme or unique case that is commonly difficult to approach (Yin 1994). Multiple-case studies are used either when the results of the earlier case study are verified, that is similar results are predicted, or when contrasting results are obtained, but for predictable reasons (Yin 1994). Furthermore, according to Easterbrook et al. (2008) case study research uses purposive sampling rather than random sampling, since the aim is to select the cases most relevant to the research proposition. Sometimes it is sufficient to identify a typical case to gain more insight into a common situation, but multiple-case studies usually provide greater validity (Easterbrook et al. 2008). In this thesis, the multiple-case studies approach is used, since the theory on continuous planning and roadmapping is not yet well formulated, and practical experiences from the field of research are difficult to find. Therefore, the purpose is to fill the gaps found in the literature, and thereafter, to create a theory relating to continuous planning and roadmapping practices and processes based on the case descriptions. To verify the theory based on the case descriptions, the experiences of several companies should be gathered and analysed.

3.3 Case selection and description

The research included three global ICT companies. These case companies were selected because they were large companies with multiple levels of planning, which enabled planning practices to be studied throughout the organisation; all the way from strategic planning through to business and financial planning to product and release planning. Also, these case companies were selected because they had transformed their organisational practices towards agile and lean software development. The companies had started with the agile method, more precisely with the Scrum method (e.g. Abrahamsson et al. 2002), and then, later on, complemented it with the lean approach (e.g. Middleton et al. 2005). These companies had also recognised the importance of continuous software development, and were moving towards the practices of continuous deployment and planning. In addition, one of the reason for selecting these case companies was they were easily accessible through VTT's (VTT Technical Research Centre of Finland Ltd) research projects, namely the Embedded Systems Engineering in Collaboration (MERLIN) project of Information Technology for European Advancement (ITEA) (2004–2007), Cloud Software Program (2010–2013), and Need for Speed (N4S) program (2014–2017) of Digile (Finnish Strategic Centre for Science, Technology and Innovation). The case companies together with the data collection methods are shown in Table 2.

Table 2. Case companies.

Case ID	Number of employees	Industry	Data collection method	Number of interviews	Role of the interviewee
A	1800	IT, products and services	Interviews, series of meetings with the case company representatives, and analysing company's internal data	1 in 2011	Head of quality and environment
				9 in 2014-2015	Scrum master, Team leader/project manager, Product manager, Sales director, President of a business segment, Vice president of a business area, Quality managers (2), Business developer

B	1000	Data security	An interview and analysing company's internal data	1 in 2011	Project manager
				12 in 2014	Product marketing manager (2), Chief strategy officer (CSO), Senior product manager (2), Product marketing management (2), Director of product management (3), Executive vice president, Vice president of R&D
C	110 000	Communi-cation equipment, software, and services	Interviews	3 in 2015	Line manager, Domain manager, Product owner

3.3.1 Case A

Case company A is a global company from Finland with roughly 1800 employees that operates in eight countries, providing cutting-edge technological solutions to the automotive and wireless industries. In the Wireless Business Segment (the case context), the company offers products and solutions based on their own platforms for defence, public safety and other authorities markets, Internet of Things (IoT) markets as well as for industrial use.

3.3.2 Case B

Case company B is an online security and privacy company from Finland with approximately 1000 employees at twenty offices around the world. The company offers millions of people around the globe the power to surf invisibly and store and share data, safe from online threats. The company was founded in 1988, and has partnerships with more than 200 operators, and it operates 22 wholly owned subsidiaries.

3.3.3 Case C

Case company C is a multinational provider of communications technology and services from Sweden. It operates in the environment of communications technology by providing equipment, software and services to its customers. The

company employs more than 110 000 people and works with customers all over the world.

3.4 Research process

The research process was initiated by conducting a literature review. Then, the empirical data in the thesis were collected in two ways: firstly, by conducting an initial inquiry consisting of both questionnaire study and semi-structured interviews, and secondly, by conducting a multiple-case study. The research process together with the main phase of the research is illustrated in Figure 9. Next, the three main phases of the research are described in more detail, and then the empirical data collection methods used during the research are defined.

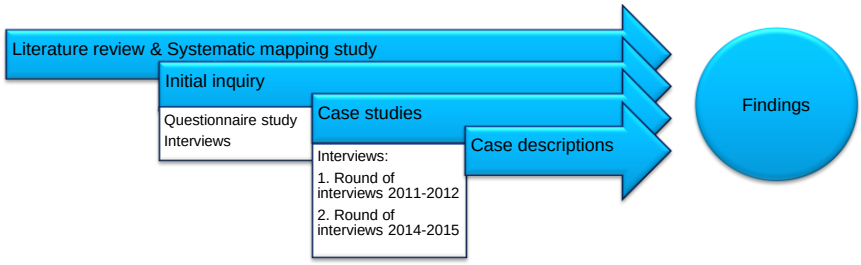


Figure 9. Research process and the main phases of the research.

3.4.1 Literature review

The research began with a literature review. The purpose of the literature review was to understand the current state and knowledge relating to the main areas of the research: roadmapping, continuous planning, and agile and lean software development (which is the scope of the research, as described in Subchapter 1.2). Also, according to Yin (2003), constructing a preliminary theory relating to the study is essential, especially, in the case study research. The development of the preliminary theory helps to define the appropriate research design and data collection as well as generalising the results of the case study research (Yin 2003). The preliminary theory of this thesis is presented in Chapter 2. The intention of the literature review was also to discover gaps in the literature and thus focus the research on the most unexplored areas of research, and then to help to structure the empirical data collection, for example, formulating the questionnaire and interview themes and questions.

The literature review was continued by conducting a systematic mapping study on continuous deployment (Kitchenham and Charters 2007, Petersen et al. 2008). The systematic mapping study was conducted in order to obtain an overview of the research on CD. Also, it was hoped to bring insights to some of the vital areas

of this research, namely to continuous planning and to agile and lean software development. The main difference between systematic mapping studies and the systematic literature review (SLR) is that while SLRs aim to 'identify best practice with respect to specific procedures, technologies, methods or tools by aggregating information from comparative studies', mapping studies focus on 'classification and thematic analysis of literature on a software engineering topic' (Kitchenham and Charters 2007). In the case of CD, although the term is frequently used in industrial and academic circles, its meaning and the main factors that are part of CD have remained undeveloped. Therefore, before aggregating information in terms of research outcomes, it was realised that there is a need to provide a comprehensive definition of CD. That is to identify, categorize and analyse the available research on the topic of CD in order to describe the phenomenon, obtain an overview of its state-of-the-art practice, determine the scientific evidence in the reported results and determine areas that are suitable for more detailed study. During the mapping study, the process of an SLR was followed as established by Kitchenham and Charters (2007), but adapted to a mapping study process, as suggested by Petersen et al. (2008). The research process is outlined in more detail in Paper V.

3.4.2 Initial inquiry

The research was started already in 2006 with an initial inquiry about product roadmapping consisting of both a questionnaire study and interviews. The framework for the initial inquiry is presented in Appendix 1. Firstly, a literature research was conducted in order to understand the state-of-the-art of product roadmapping. In addition, the purpose of the literature review was to focus research on a few of the most important issues from the field of research, and to prepare the proposed questionnaire based on the literature review. The data collection was carried out, firstly, in the form of a questionnaire study to collect background knowledge on the companies on the application of product roadmapping. The interviews were then conducted to collect more in-depth knowledge about product roadmapping in different companies. The process of the initial inquiry research was carried out in five phases: 1) formulating the questionnaire, 2) pilot testing of the questionnaire, 3) analysis of the questionnaire results, 4) interviews, and 5) analysis of the findings.

The empirical research of the initial inquiry was carried out as qualitative research conducted with a questionnaire-based survey study (Oppenheim 1992) and semi-structured interviews (Järvinen 2001). In the questionnaire-based survey method, Alreck and Settle (1995) classify three methods of data collection: personal interviewing, telephone interviewing and mail data collection, all of which were used during the research. In selecting the respondent organisations for the questionnaire survey and interviewees, purposive sampling (Nardi 2003) was used. Purposive sampling involves designating a group of people for selection because they have some traits that are important for the study (Nardi 2003). The

questionnaire respondents were selected based on VTT's electronic mailing lists. Also, the questionnaire was sent to the European-wide MERLIN research projects' partner companies, since they were assumed to have knowledge of the field of study. Then the interviewees were selected based on the respondent's experience in product roadmapping. The emphasis of the empirical research was put on the interviews. The questionnaire was mainly used to obtain the right persons to be interviewed, but also to acquire data about product roadmapping.

The inquiry form of the questionnaire study was divided into three parts: general information, company profile and product roadmapping process. The questions were phrased based on the literature (e.g. Tabrizi and Walleigh 1997, Albright 2002, McCarthy 2003, Rautiainen et al. 2003, Phaal et al. 2003, Lehtola et al. 2005, Groenveld 2007, Van de Weerd et al. 2010). In more detail, the questions related to the product roadmapping, collaborative development, and requirements prioritisation. All the questions in the questionnaire study were asked in a structured form so that the respondent could select from several alternatives. However, in order to obtain more information about the research areas, each question could also be replied to with an open answer. Thus, the questionnaire study included both structured and unstructured questions. The structured questions were selected in order to shorten the response time and therefore to receive more replies. Also, the structured questions were chosen to make the analysis easier and the conclusions stronger.

The questions for the inquiry form were planned and arranged carefully in advance so as to produce the right form of questions and to avoid misunderstandings. The questionnaire was pre-tested inside VTT among researchers, and based on the pre-test results, small modifications to the questionnaire were made. After the pilot testing of the questionnaire, in the summer of 2006, the questionnaire was sent to potentially interested contacts by using purposive sampling (Nardi 2003), that is to companies assumed to have experience of and interest in product roadmapping. An electronic mailing list was prepared, and the questionnaire was emailed to over 600 respondents. The high response rate can be partly explained by the fact the mailing list was not ranked, for example, according to respondents' roles, thus the questionnaire was also sent to those respondents who were not actually involved with the scope of research. As a result, a total of 59 responses were received, and seven of the respondents lacked experience in product roadmapping and were hence unable to complete the form. Though the resulting response rate of the questionnaire was quite low (10%), the 52 replies from 34 different companies can be claimed to provide a wide enough perspective for further analysis, especially for the selection of people to be interviewed. One of the replies, however, was excluded from the analysis due to an incompletely filled questionnaire form. The questionnaire respondent companies originated from Finland, Sweden and the Netherlands. All companies were involved in software products or service development in the field of ICT industry. Their scopes varied from own product development to the development of components for external partners. The size of the case companies, measured as the number of overall employees, was distributed among the given categories

in the questionnaire (under 10 employees, 10–49 employees, 50–250 employees, and over 250 employees) with an emphasis on medium and large companies.

After the questionnaire study, the interviews were planned. The purpose was to gain in-depth knowledge of how product roadmapping was conducted in the different companies. The interviews were semi-structured, since the interviews included structured questions and proceeded according to certain vital themes of the research. The vital themes of the research were selected based on the literature analysis and findings of the questionnaire studies. Based on the literature, the themes were related to roadmapping, collaborative development, and requirements prioritisation. Additionally, based on the findings of the questionnaire studies, the themes related to collaborative viewpoints to product roadmapping and different phases of product roadmapping. Therefore, especially the company experiences of creating product roadmaps in inter-company collaborations were emphasised during the interviews. Additionally, questions were asked relating to the benefits and problems of the product roadmapping. As is characteristic of semi-structured interviews, the themes were the same for all the interviewees, but the questions varied between the different interview sessions. Equally, the interview questions were partly planned in advance, but not phrased or arranged in detail. Moreover, the intention in the interviews was to emphasize the interviewee's experiences and their own opinions on the field of study. Furthermore, the interviews could be considered focused interviews, since the respondents were interviewed personally for a short period of time, i.e. not more than an hour (Yin 1994). Altogether, nine people representing eight different companies were selected to be interviewed. Two of the interviews were face-to-face, and the other seven were conducted by phone. More detailed information about the interviews conducted is presented in Appendix 3, and a list of interview profiles is presented in Appendix 4.

3.4.3 Multiple-case study

After the first phase of the research (that is the initial inquiry) as previously described, the actual case studies were conducted. The theme of the research was broadened from product roadmapping to continuous planning (in which product roadmapping was seen as part of the larger context). The research was conducted as empirical research in which three case companies were selected. In selecting the case companies, a purposeful sampling approach was used (Nardi 2003), similar to the initial inquiry. As explained, purposive sampling involves designating a group of companies, projects or people for selection since they have some traits that are important for the study (Nardi 2003). These case companies were selected because they were large global ICT companies with multiple levels of planning, and they had adopted agile and lean software development practices.

The main data collection method was interviews, which were either semi-structured or narrative by nature. The list of interviews conducted for this research is presented in Appendix 3. The semi-structured interviews were conducted similar

to the interviews in the initial inquiry. For example, the interviews included open-ended questions (Runeson and Höst 2009) and dealt with certain vital themes of the research. The vital themes of research were created based on the literature review of roadmapping and continuous planning, as well as the findings of the initial inquiry. The interview protocol was organised in four pre-defined themes, but it allowed for openness and flexibility within the themes. The themes were 1) definition and background of continuous planning (including planning practices in general), 2) continuous planning process (including levels of planning and time frames), 3) participants of continuous planning, and 4) lessons learned (including benefits and barriers of continuous planning as experienced). The vital themes of the research together with the research questions are presented in more detail in Appendix 2. The themes were the same for all the interviewees, but the questions varied between the different interview sessions. The interview questions were partly planned in advance, but not in detailed phrasing or arrangement. The interviews could also be considered to be focused interviews, since the respondents were interviewed personally for a short period of time, ranging from one to two hours (Yin 1994). In total, 26 interviews were conducted. More detailed information about the interviews conducted is presented in Appendix 3, and a list of interview profiles is presented in Appendix 4.

The interviews to the case companies were held in two different rounds and research programmes. The first round was conducted in the Cloud software research programme during the years 2011–2012, and the second round of interviews was conducted in the N4S research programme during the years 2014–2015. During the first round, representatives of two case companies (Case A and Case B) were interviewed. After these interviews, it was noticed by the researcher that the interviewees were selected based on their experience of continuous planning, and thus the interviewees were from a totally different part of the organisation. Therefore, it was clear that a wider perspective on continuous planning practices was needed and that people from different levels of the organisation should be interviewed in order for the researcher to understand continuous planning throughout the organisation and also to be able compare the interview results between the case companies and to draw conclusions.

In the second round of interviews, semi-structured interviews were conducted in Case A and in Case C. At the same time as these interviews, the researcher also sought to use narrative interviews in Case B, to discover how the terms and practices of continuous planning would emerge from the interviews without describing these to the interviewees. This was partly because the term of continuous planning was quite new and not commonly known among the interviewees. Further, the researcher did not wish to mislead the interviewees in any way; instead, the researcher encouraged interviewees to discuss the company's planning practices openly. The interviewees were allowed to tell their story from their own perspective and in their own words and ways of expression without a predefined list of interview questions or structured interview agenda. The same vital themes of the research as in semi-structured interviews were used during the narrative interviews to guide the interviews, but without the list of

predefined questions. The research process for each case company that was involved in this research is described in more detail hereafter.

In **Case company A** ten interviews were conducted altogether. The research data from Case A began being collected during 2011–2012 via a series of meetings and one interview as well as by analysing company specific material. In 2011, eight meetings were held altogether to exchange information about continuous planning, in which three people were involved: two researchers from VTT and one representative of the case company. During the meetings, for example, the vital elements of continuous planning were discussed and drafts of the continuous planning framework were drawn up, forming the vital elements of continuous planning (presented in more detail in Paper III). Then, in 2012, one semi-structured interview was held to clarify how continuous planning is conducted and to discover how the information flows up and down between the different levels of the organisation. The interview was recorded and subsequently transcribed by an external consultant company. A summary report from the interview was written based on the transcript and with the help of analysing company specific material (including PowerPoint slides). The summary report was commented on and corrected by the interviewee in order to validate the correctness of the data. The interviews were continued from October 2014 until June 2015. Altogether nine interviews were held involving various people with different roles (for example the scrum master, project manager, product manager, sales directors, president of a business segment, quality managers, business developer, and director of defence). All these interviews were semi-structured, having a mixture of predefined and free-form structure, and the interviews dealt with the vital themes of the research. The interviews were held in Finnish, and each interview lasted for one and a half hours. A minimum of two researchers were present at the interviews so as to ensure that the relevant information was correctly extracted. All the researchers participating took notes during the interviews, and notes were compared after the interviews so as to ensure consistency. The interview was recorded and then transcribed by an external consultant company. Also, the researchers were provided with internal company materials (e.g. PowerPoint slides) by the company representatives to be used in the analysis.

In **Case company B** we conducted thirteen interviews altogether. The research data for Case B began being collected in 2011 via a semi-structured interview and by analysing the company's internal data. After the interview, the case company's internal memos of continuous planning were analysed. The data included information relating to continuous planning practices which were already discussed during the interview, but were considered vital in order to clarify and verify the practices and processes related to continuous planning. The interview was organised as a semi-structured interview that proceeded along the vital themes of the research. The interview lasted one hour and 40 minutes and involved one interviewee and two interviewers. Thereafter, the interview was transcribed and analysed. After the interview, internal memos pertaining to continuous planning at the case company were analysed. This data included

information related to continuous planning practices that were discussed during the interview, although this information was nonetheless considered vital in terms of clarifying and verifying the practices and processes of the company related to continuous planning. The research was continued up to October 2014, when twelve interviews were conducted altogether. The interviews involved people from various roles (e.g. senior product marketing managers, product marketing managers, senior product managers, directors, an executive vice president, a chief strategy officer, and a vice president of R&D). These interviews were narrative by nature, proceeded along the vital theme of research ('customer insight in planning'). Two researchers were present in all the interviews, and both of the participating researchers took notes during the interviews. Notes were compared after the interviews to ensure consistency. The interviews were held in Finnish or English and each interview lasted from one hour to one and a half hours. Also, all the interviews were recorded and transcribed by an external consultant company afterwards.

In **Case company C** altogether three interviews were conducted in January 2015. The interviewees represented the following roles: line manager, domain manager, and product owner, from one of the company's business areas. The interviews were semi-structured and dealt with the vital themes of research. The interviews were held in Finnish and each interview lasted from one to one and a half hours. During the interviews, there were two researchers sharing the responsibility; one of us asked the questions and one took notes. In addition to the interview notes, all the interviews were recorded and then transcribed by an external consultant company.

3.4.4 Empirical data collection methods

In this thesis, data collection was carried out in the form of a questionnaire study and interviews, which pertain to the chosen case study research approach (Järvinen 2001). These data collection methods were selected because with a questionnaire study, the basic knowledge in companies using product roadmapping could be revealed, and with interviews, more in-depth knowledge about product roadmapping and continuous planning could be discovered. The questionnaire study is considered an appropriate method when the number of matters to be inquired into is relatively small and the number of respondents is relatively large, as in this research. The questionnaire studies were extended with interviews in order to specify the received information from the respondent. Additionally, interviews were conducted, since they were expected to bring out new aspects that would not be otherwise revealed (Järvinen 2001).

In a questionnaire study, the data were collected with an inquiry form. This inquiry in a paper or electronic format contains a set of structured or unstructured (that is open) questions intended to be answered by the people selected. Structured questions are used when the subject of the questions holds a generally accepted classification that is extensive. Hence, structured questions are used in

theory-testing studies. In contrast, unstructured questions are used when the subject of the question is not yet structured. In that case, the questions are expected to reveal experiences from practice. Thus, these questions are used in theory-creating studies (Järvinen 2001). These phrasings of questions also apply to interview studies.

In an interview, data are collected in a discussion between the interviewer and interviewee, in which the purpose is to gather certain information from the interviewee (Järvinen 2012). There are three types of interviews: structured, semi-structured, and unstructured (e.g. Denzin and Lincoln 2000, Järvinen et al. 2014). The interview type depends on the research approach used (Järvinen 2012) and on the advance planning of the interviews (Järvinen 2001). In the structured interview, questions are carefully planned and formulated before the interview, based on the research framework and hypotheses. In an unstructured interview, the themes for research guide the interview (Järvinen 2001). These interviews are not planned in detail beforehand, and thus the interviewees are asked open-ended questions. Additionally, the interviewees can be asked for the facts of a matter as well as for their opinions about events (Yin 1994). The semi-structured interview includes both structured questions and open themes of discussion (Järvinen 2001). A narrative interview is one type of open, unstructured interview. Narrative interviews concern the production of stories that people tell spontaneously in interview situations (Eriksson and Kovalainen 2008). Accordingly, a narrative interview is open in two different ways: there is no prior hypothesis to be tested and the interviewee is encouraged to talk openly. The interviewee is allowed to talk about their study from their own perspective and in their own words without a predefined list of interview questions or a structured interview agenda (Eriksson and Kovalainen 2008). In a narrative interview, open-ended can either cover a longer period of time or focus on a specific event. The questions are posed without defining the content of what the story should be about, letting the interviewee decide that; hence the interviewee is given the freedom to speak uninterrupted (Eriksson and Kovalainen 2008).

In this research, the interviews were both semi-structured and narrative in nature. The semi-structured interviews included structured questions, and proceeded along certain vital themes of the research. The vital themes of research were created based on the literature review and findings of the questionnaire study. The themes of the research interviews are listed in Appendix 2. The interview themes were the same for all the interviewees, but the questions varied between the different interview sessions. Additionally, the interview questions were partly planned in advance, but not with detailed phrasing or arrangement. Moreover, the intention in the interviews was to emphasise the interviewee's experiences and their own opinions of the field of study. The narrative interviews were held to find out whether continuous planning would emerge from the interviews without describing it to the interviewees in advance. The interviewees were allowed to tell their story from their own perspective and in their own words without a predefined list of interview questions, as in the semi-structured interviews.

3.5 Data analysis

Data analysis consists of examining, categorizing, tabulating or recombining the research evidence in order to address the initial research proposition (Yin 1994). According to Yin (1994), the analysis of case study evidence should start with a general analytic strategy yielding priorities for what to analyse and why. Within such a strategy, four dominant analytic techniques could be used: pattern-matching, explanation building, time-series analysis, and programme logic models (Yin 1994). The ultimate goal is to treat the research evidence fairly, to produce compelling analytic conclusions, and to rule out alternate interpretations. The role of the general strategy is to help the researcher choose among different techniques and to complete the data analysis successfully (Yin 1994). Yin (1994) proposes two general strategies for analysing case studies: relying on theoretical propositions and developing a case description. Developing case descriptions from each case was the main analytical strategy applied in this thesis. Firstly, a theoretical proposition was created (that is the literature review) which led to the case studies. Based on the literature review, it was discovered that empirical experiences from the field of research were difficult to find, and thus empirical case research is needed. The case studies in that sense reflect the set of research questions, reviews of the literature, and new insights (Yin 1994). Secondly, a descriptive framework for organising the case studies was formed by creating a case description from each of the cases.

During the data analysis, the generic process of data analysis presented by Creswell (2003) was used both to analyse the data from the questionnaire study and the interviews. Accordingly, the survey design begins with a discussion about the purpose of the survey, the identification of the population and sample for the study, the survey instruments to be used, the relationships between the variables, the research questions, specific items of the survey and the steps to be taken (Creswell 2003).

During the analysis of the questionnaire data, the information was presented as a series of steps, which including the following information: response rate, response bias (that is the effect of nonresponses of survey estimates), descriptive analysis of data for all independent and dependent variables in the study, developing scales and mentioning reliability checks for the scales, and identifying the statistics. The questionnaire study provided a numeric description of trends, attitudes and opinions for the sample population. From these sample results, generalisations and claims about the population could be made. However, the main intention of the questionnaire study was to reach the people to be interviewed to acquire more detailed information about the roadmapping practices used from the case companies.

All the interviews were digitally recorded as they were taking place and notes were taken by the interviewee so that the responses could be verified afterwards in order to obtain the correct information. After each interview, the digital recording was transcribed by an external consultant company. Subsequently, all the

transcribed interview data were analysed with the help of a qualitative data analysis tool called NVivo. During the data analysis, all transcribed data were carefully read through in order to obtain a general sense of the data and to identify recurring elements and concepts, as is common in open coding techniques (Strauss et al. 1998). All the interviews were analysed using the generic process of data analysis presented by Creswell (2003). A coding process was used to categorise the data and label the categories. As with grounded theory (Strauss et al. 1998), the researchers let coding categories and relationships emerge from the data. Therefore, all interviews were read several times, and after going back and forth in the empirical data, and applying the process of constant comparison, the researchers reached a state in which no new significant categories or concepts emerged, which is called 'theoretical saturation' by (Strauss et al. 1998). The purpose of the coding process was also to generate descriptions for the categories to generate in turn a small number of themes that represented the major findings of the qualitative data. These descriptions and themes were then presented in qualitative narratives in case descriptions (Creswell 2003). In addition to the interviews, company-specific material including internal memos, documents and slides and all corporate websites and brochures were used to complement the case descriptions. The case descriptions were validated by the interviewees by reading through them and making corrections if needed. Finally, based on the empirical research, the current status of continuous planning at each case company was decided upon. All the case descriptions were the result of our data analysis and they reflect perceptions, experiences and beliefs as held by the interviewees involved in the study.

4. Original publications

In this chapter, the original publications included in the thesis are introduced. The purpose is to present the overall view of the papers and their key contributions. Altogether, five publications are included. The topics of the publications are the following:

I Challenges for product roadmapping in inter-company collaboration

II Software product roadmapping in a volatile business environment

III Continuous planning: an important aspect of agile and lean development

IV Defining continuous planning through a multiple-case study

V Continuous deployment of software intensive products and services: a systematic mapping study.

Each publication gives answers or clarifies the research questions of the thesis from different perspectives. Papers I and II describe the current state of roadmapping, thus contributing to RQ1: How is roadmapping conducted in software development? For example, by providing a definition for a roadmap and roadmapping, identifying the process and its main participants in software development context. Paper I also takes into account the collaboration perspective for creating product roadmaps. Papers III and IV contribute to RQ2: How is continuous planning conducted through roadmapping in agile and lean software development? As an example, a response may be by providing a definition for continuous planning in an agile and lean software development context. Furthermore, Paper III identifies the vital elements of continuous planning, and both Papers III and IV define the main levels of planning and their time frames. Paper V contributes to RQ3: How is continuous planning conducted in CD-driven software development? It describes the recent evolution of software development towards CD and brings out both the benefits and challenges associated with it. In the chapters that now follow, the focus and central conclusions of each paper are defined in more detail.

4.1 Paper I: Challenges for product roadmapping in inter-company collaboration

The paper focuses on product roadmapping, and highlights that it is a critical activity in product development, since it provides a link between business aspects and requirements engineering, and thus helps to manage a high-level view of the company's products. The paper points out that inter-company collaboration, such as joint R&D partnerships, customer–supplier relationships (including outsourcing), and technology exchange agreements and licensing are a common way of developing software products. Through collaboration, organisations gain advantages such as flexibility with in-house resources, savings in product development costs and establish a physical presence in important markets. The paper gives an overview of product roadmapping and then presents current industry practices based on empirical research (presented in Subchapter 3.4.2 Initial inquiry). The paper presents key challenges to and opportunities for creating product roadmaps through collaboration, including for example, the most important activities to consider, the most typical problems, and how those problems can be avoided.

Based on the findings given in the paper, creating a product roadmap in inter-company collaboration depends on the product to be developed and on the form of co-operation. Product roadmapping in inter-company collaboration is affected by the following aspects: the period of the product's life span, the closeness of the relationships and the type of partnership, i.e. who is in control of the activities taking place. In a customer–supplier relationship, such as outsourcing, the central idea in creating product roadmaps together is to create a mutual understanding and ensure the confidentiality of the roadmap. In joint R&D partnerships, confidentiality was seen as important. However, the major difference compared to the customer–supplier relationship was that one of the joint R&D partners is the leading partner, who had the overall idea of the product to be developed and was most likely the owner of the roadmap. In technology exchange or licensing agreements, the relationship was considered as more of a matter of legal agreements and contracts, and thus the roadmaps were not shared so freely as in the other collaboration modes. Furthermore, the findings indicate that continuous communication is important and, especially, at the beginning of the product roadmapping process, face-to-face meetings are essential to avoid misunderstandings. Also, openness between collaboration partners is important in order to mutually share ideas and views. However, knowledge should be shared without losing critical confidentiality. Hence, creating good and confidential relationships with partners, as well as creating long-lasting customer relationships, are of major importance. Instead, project management is challenging in a collaboration situation, because in the case of multiple actors in the product roadmapping process, problems are caused when dividing tasks between partners. Therefore, the work should be transparent in order to avoid duplicate work and grey areas.

4.2 Paper II: Software product roadmapping in a volatile business environment

The paper focuses on roadmapping from the software development perspective. The main goal of the paper is to increase the current empirical evidence on product roadmapping by defining the main stakeholders and their roles during the process, organising the product roadmapping process, establishing the main benefits and challenges faced during the process and identifying the most critical phases of the process. The empirical evidence is based on both quantitative and qualitative data (presented in Subchapter 3.4.2 Initial inquiry). The paper presents a research framework for software product roadmapping, which is created based on the existing literature, so as to help structure and present the research results. The main elements of the framework are as follows: stakeholders, process phases and the perceived impact of product roadmapping.

Based on the results given in the paper, organisations view the roadmap mainly as a tool for strategic decision-making, as their aim is to show the future directions of the company's products. However, only a few organisations appear to have an explicit approach to handling the mechanisms for creating and maintaining such a roadmap. Based on the key findings of the paper, product roadmapping is a continuous process, since the roadmapping team has regular meetings (for example biweekly, quarterly, or biannually) in order to create, update or review roadmaps. The product roadmapping team consists of several stakeholders. At least the following stakeholders are seen as important in the process: product management, marketing, customer and partner representatives, and development. Also, based on the empirical findings, feature management is seen as being the key aspect in product roadmapping and accordingly the product roadmapping process is proposed to consist of the following phases: capturing features, analysing features, prioritising features, roadmap validation and agreement, and change management of the roadmap. The most critical phase of product roadmapping is prioritising features. Similarly, the most problematic areas of product roadmapping are prioritising features, managing changes and maintaining roadmaps, as well as sharing information, communication, and creating a roadmap agreement. Finally, it is suggested that the strategic importance of product roadmapping is likely to increase in the future and, as a new type of agility is required in order to survive in the turbulent and competitive software business environment.

4.3 Paper III: Continuous planning: an important aspect of agile and lean development

The paper focuses on continuous planning, which is seen a relatively new and not yet fully studied field of research, especially from the perspective of agile and lean development organisations. To augment knowledge in this field, the paper presents both a literature review and empirical findings from case studies (given in

Subchapter 3.4.3 Multiple-case study), which reveal how companies view and conduct continuous planning. The findings given in the paper highlight the importance of continuous planning throughout an entire organisation, including in regards to the elements of continuous planning: organisational planning, strategic planning and business planning, and their close interrelation. Organisational planning serves to define a plan's organisational level and its time frames; strategic planning serves to set an overall plan for an organisation, and business planning serves to establish the budgeting frame of a plan. Furthermore, the paper points out that continuous planning is not currently conducted throughout an entire organisation and may only engage a certain level of planning. While continuous planning in agile and lean organisations commonly relates to release planning, the paper sheds light on a broader perspective than this by defining continuous strategic and financial planning, as well as continuous project- and team-level planning. Based on the empirical findings of the paper, the majority of long-term plans looked three years ahead. At the strategy, business, or project levels, plans were reviewed quarterly, whereas at the team level plans were reviewed biweekly or weekly. While continuous planning was mainly understood as short-term planning, this did not remove the need for long-term planning, as strategic, business, market and portfolio planning had to be constantly considered at the higher levels of each organisation.

The paper also presents some background arguments for adopting continuous planning. The motivation towards continuous planning has arisen from both external and internal challenges that companies face in today's volatile market environments. There is a clear need for continuous planning, as organisations face difficulties in developing long-term plans due to constant changes in their customer and market-bases, as well as in product and technology development. Moreover, recent financial crises have caused companies to rethink their approaches to planning and to realise the importance of continuous planning both from an operational and financial perspective. Therefore, it is suggested that the importance of continuous planning will only increase dramatically in turbulent business environments that include ever shorter planning cycles and the need to improve transparency and knowledge-sharing in organisations.

4.4 Paper IV: Defining continuous planning through a multiple-case study

The paper focuses on continuous planning throughout the organisation. It is believed that continuity is required at all levels of an organisation, from business strategy and planning to software development and operational deployment, as well as between these levels. Continuous planning is seen as one of these activities. The paper presents empirical evidence from a multiple-case study (presented in Subchapter 3.4.3 Multiple-case study) in which the various levels of planning, along with their time frames, are explored. The paper described how planning is currently conducted in three different organisations. It describes the

main levels of planning: strategic, financial, business, product and team, as well as their time frames.

The key findings of the paper highlight that none of the case companies utilised the practices of continuous planning throughout the organisation. Furthermore, the continuity of the activities was often based on conditionality, for example, on whether the circumstances forced companies to update and review their plans. The continuity of the activities was explained by both internal and external changes in the companies. External changes, such as, the turbulent business environment, forced the case companies to adopt continuous strategic and financial planning practices. Internal changes, such as, the adoption of agile and lean development practices, forced all the case companies to shorten their product planning review cycles to months and team-level planning to weeks or days. Business planning, in contrast, was the only activity that was not seen as continuous in any of the case companies. In a similar way, regarding strategic planning, continuity might also be needed in business planning in order to respond and react to changes in the business environment. Both business and product planning should be continuous and proactive, rather than reactive, in nature. With the help of continuous business and product planning, companies may be able to influence the markets by inventing and developing new products and services, as well as being able to react to changing market requirements.

4.5 Paper V: Continuous deployment of software intensive products and services: a systematic mapping study

The paper focuses on continuous deployment, as it is realised that software-intensive industries are moving towards the adoption of a value-driven and adaptive real-time business paradigm. Therefore, the traditional view of software as an item that evolves through releases every few months is being replaced by continuous evolution of software functionality. The paper presents results from a systematic mapping study (presented in Subchapter 3.4.1 Literature review). It classifies and analyses literature related to continuous deployment in the software domain in order to scope the phenomenon, provide an overview of its state-of—the-art, investigate the scientific evidence in the reported results, as well as identify areas that are suitable for further research. As a result of the analysis, altogether 50 primary studies published between 2001 and 2014 were identified. An in-depth analysis of the primary studies revealed 10 recurrent themes (or factors as referred to in the paper) that characterise continuous deployment. These themes include: 1. fast and frequent release, 2. flexible product design and architecture, 3. continuous testing and quality assurance, 4. automation, 5. configuration management, 6. customer involvement, 7. continuous and rapid experimentation, 8. post-deployment activities, 9. agile and lean software development, and 10. organisational factors, including integrated corporate functions, transparency and an innovative and experimental organisational culture. The most relevant themes for this thesis are themes 1 and 9.

Based on the key findings of the paper, continuous deployment goes beyond agile and lean software development; thus, agile and lean software development methods and practices are the first steps the organisation can take toward CD. Hence, ASD methods and practices can be considered as an enabler for continuous deployment. Also, CD scales ASD practices throughout the whole organisation instead of focusing only on team-level activities. CD also transforms traditional ASD practices and methods into a continuous flow. Similarly, almost all of the primary studies make reference in one way or another to accelerating the release cycle by shortening the release cadence and turning it into a continuous flow. The fast and frequent release means the ability to release software whenever the organisation wishes, based on need, which may be weekly or daily. CD requires that the planning activities should be done more frequently so as to ensure alignment between the needs of the business context and software development, as well as requiring tighter integration between planning and execution. Continuous planning is discussed and presented in the paper as one of the mechanisms to achieve faster release cycle. The paper also presents a set of benefits and challenges for CD that are partly associated with continuous planning. One of the most immediate benefits of applying CD is shorter time-to-market through fast and frequent releases. For instance, it shortened their delivery cycles from months or weeks to a continuous flow or daily deliveries. Shorter release cycles enable companies to constantly develop, learn and improve their offering based on instant customer feedback and thus, companies can quickly learn what customers value and so focus on deploying relevant functionalities that meet customers' expectations. However, transformation towards CD is an evolutionary process and requires investment in deployment processes, as well as changes in people's mind-set and organisations' way of working. Furthermore, CD also requires a change in organisational culture, buy-in from all key stakeholders, and transparency in the organisation.

5. Discussion and conclusions

In this chapter, the main findings of this research are discussed and the concluding words are provided. The chapter is divided into three subchapters. Firstly, a summary of the research is presented. The research questions and the main research problem posed at the beginning of this study are answered with respect to the empirical findings given in the original publications. Secondly, the research results are evaluated. The main theoretical contributions and empirical implications are presented. Then, the reliability and validity of the research as well as limitations of the results are discussed. Thirdly, the conclusions are set out and recommendations for further work are discussed.

5.1 Summary of the research

In what follows, the answers to research questions are provided. All three sub-questions are addressed separately based on the individual findings of the original research papers. Then, the answer to the main research problem of this thesis is given.

RQ1. How is roadmapping conducted in software development?

A roadmap is a plan or a leading map of the company's future directions. A roadmap also provides a high-level understanding of scoping the strategy, which allows better planning and commitment to the set plans. The roadmap *structure* is commonly presented as a multi-layered time-based chart. The structure constitutes the levels of planning, and the time frame of a plan. In a software development context, the roadmap typically relates to products. A product roadmap helps to structure and arrange product development in order to know how to use certain resources and what is to be done and when. Hence, product development is somehow deterministic and enables steering of the product implementation. With a roadmap, tasks to be done can be prioritised, and thus resources can be allocated to the most profitable projects. It can even be verified from the roadmap that the right things are being done at the right time. Additionally, in software development with a good roadmap, the customers' needs

can be met with a product that they really want, and hence providing a competitive advantage. Also, a roadmap is a central tool for communication, and therefore it should be shown to the company's own staff as well as to partners. It gives a clear idea of what is about to be done, and enables communication about forthcoming strategic projects. Furthermore, in a collaboration situation, a roadmap is one of the main documents describing what the parties have agreed to and what is about to be done together, so that everybody knows the goals. From a roadmap, the collaboration partners can see what others are currently doing and which phase they should be in. Thus, a roadmap simplifies the synchronisation between collaboration parties and also provides vigour, backbone and predictability of product development for the partners.

Roadmapping describes the process of creating and revising roadmaps. Roadmapping improves predictability, and hence reduces the occurrence of surprises during the development. As roadmapping is a central method for communication, it gives a clear idea of what is about to be done, and enables communication in regards to forthcoming strategic projects. Roadmapping is a continuous process in which roadmaps are updated and reviewed. The *roadmapping process* is conducted by the roadmapping team which has meetings biweekly, quarterly, or biannually, for instance. In addition, the product roadmapping process commonly begins with customer requirements. These requirements can, for example, be proposals for improvement or new product features, as well as the customer's goals or expectations. Furthermore, the requirements can also come from the company's internal research unit or through competitor analysis.

In a software development context, the most important *participants* in the product roadmapping process are: product management, marketing, customer and partner representatives, and development including manufacturing and engineering. Customer and partner representatives are the external participants to be involved in the process. The participants in the roadmapping process have two basic *roles*: a member of the roadmapping team and an owner of the roadmap. The owner's role is considered to be the most important, as this person has the overall idea of the desired roadmap, and there should be someone responsible for the roadmap. The role of the owner is also to include the collection of input to the roadmap, holding the roadmap together, making the changes needed to the roadmaps, and taking care of the information flow both within and outside the company. On the other hand, the stakeholders of the roadmapping team are to bring input from different viewpoints to the roadmap, for example, to schedules and to product features.

RQ2. How is continuous planning conducted through roadmapping in agile and lean software development?

Continuous planning involves creating and revising plans as needed, typically more often than once a year. It is about implementing planning practices continuously based on need, instead of on predefined and regular planning

occasions. Continuous planning reacts to environmental changes; thus, internal and external changes trigger planning, and not only the predefined planning rhythm. The time frame of planning varies between hours and months depending on the level of planning. In terms of agile and lean software development, continuous planning refers to the organisational capacity to conduct planning in rapid cycles, in hours, days, weeks, or months. Continuous planning is not only a project- or team-level activity as described in the current literature (e.g. Shalloway et al. 2009). Rather, it involves higher-level planning as well as strategic planning, for instance. The empirical findings highlight the importance of continuous planning throughout an entire organisation. To this end, *the vital elements of continuous planning* are presented as follows: organisational planning, strategic planning, and business planning, and their close interrelation. Organisational planning serves to define a plan's organisational level and its time frames (that is the roadmap structure); strategic planning serves to set an overall plan of an organisation, and business planning serves to establish the budgeting frame of a plan.

In the light of the empirical findings based on the multiple-case study, the following levels of planning are identified: strategic planning, financial planning, business planning, product planning, and release planning. The main participants in these planning activities are respectively top management, financial management, business management, product management, and the R&D team. These various levels of continuous planning together with the main planning activities are briefly described next.

Continuous strategic planning means that the strategy process is constantly rolling, that it is continuously planned and updated based on the market and customer demand. Strategic planning is conducted annually by the top management of the company, involving people such as a chief strategy officer, a chief executive officer, president of the business area, vice presidents of the business areas, head of sales and marketing, operations and engineering. The strategy plans are reviewed and updated quarterly, or when needed by the top management. In the quarterly strategy review, the following aspects are checked: the current status, the latest updates to the strategy and the progression of the strategy deployment. After the strategy review, the results are re-reviewed and analysed by the company's business areas, who explained how each business area has progressed with the goals set in the strategy plan; where they have lagged behind, and what they have finished or left unfinished. The main phases of the strategy process are as follows. The first phase of the strategy process is the *strategy review*. The main action of this phase is conducting *a strategy health check*. During this period of time, the current status of the strategy is reviewed, and the needs for major strategy updates (such as establishing a new business area) are addressed. It involves also identification, discussion and analysis of future possibilities, referred to as strategic themes or issues that might be relevant to the company in ten years, but which should be acknowledged already now. Subsequently, the strategic work is considered in relation to strategy elements initiated that need to be updated, and plans are made regarding how to proceed with the strategy, especially in terms of changed elements. The second phase is

strategy development and acceptance. One of the main actions in this phase is creating a vision of the company's strategy and its direction. It involves long-term roadmapping both for strategic and financial planning within the time frame of the next three years, plus the current year. The company's strategy is at all times understood as a whole, yet its various elements can be changed at any time. This phase also involves an activity called *the strategy creation and elements update*, in which changes to the strategy are reviewed. Then, new strategy element updates are approved, which is then followed by the top management's strategy acceptance. This launches the third phase of the strategy process: *strategy communication and execution*. The updated strategy is communicated to the employees and strategy actions are executed.

Continuous financial planning means that there is a continuously available financial forecast. The financial frame is created once a year by the financial management. The frame is commonly created either at the end of each year (in November) or at the beginning of the year (in January) and includes a budget overview of the rest of the following year. The idea of continuous financial planning is constant in that actual expenses are continuously considered and compared to budgets. Thus, financial actions involve each of the budget items in the short-term and initiatives for the long-term. The budget plan is created annually and thereafter validated and analysed with a rolling review process. The annual budget plan is first presented to the business management who decide how the budget is divided into product domains. Then, the product management operates according to the plan and the R&D team spends the money allocated for its implementation. The budget plan is reviewed bi-annually by the business areas and quarterly by the product management in order to check its current status. Minor changes to the budget plan can be made by the product management, for example, by transferring budget to a previous or forthcoming quarter, or between the product domains in the event that some product domain does not use all the money allocated to them. That is because the budget allocated to product domains is easier to redirect than to increase. Major changes to budget plans requiring a change in the whole business area are more difficult to make and require approval from the financial management.

Continuous business planning means that the various plans relating to business management are continuously available and up-to-date as well as synchronised together. Business planning is conducted by the business management involving, for example, vice presidents of the business area. Business planning involves creating a business area strategy plan, which is carried out together with business area teams, which may include, for instance, also people from product management, product development, and research. The business area strategy plan is quite strongly product- and technology-focused; therefore, it is also known as a technology roadmap. It defines the business area's offering, meaning the products and their development, and how they are implemented. However, it is a high level document without direct links to requirements. The long-term time frame for the business area strategy plan is around three years. The first year is more accurate than the other two years. It is

created annually and updated with a monthly review cycle. The monthly meetings involve the business area team, and representatives from marketing and operations. The discussions in the review relate to the business area and its current and forthcoming products.

Commonly, company's business areas are divided into different product domains, which together form the company's product portfolio. Each product domain owner is responsible for creating plans for their own domain. The time frame for the *product domain planning* is the same as with the product planning; it has a one-year time frame with half of the year being more accurate and the rest of the year having more tentative content. The product domain plan is updated and input is collected through a quarterly review process. Also, the product domain plans are discussed and input is collected through weekly *business area steering group meetings* held by the business management, as well as through weekly *product portfolio meetings* held by the product management and R&D team. At the business area steering group meetings, the company's product portfolio is analysed; thus, the meeting is for creating a business update and ascertaining that all business areas are in sync. The meetings are held more to share information based on the product roadmaps (created by the product management) rather than to make actual decisions. For example, issues relating to product portfolios, market signals, business status, turnover, and number of subscribers, are discussed and followed up. During the weekly product portfolio meetings, the purpose is to check that the team is addressing the right issues and that the issues are in line with the product domain or product portfolio plan. Also, it reviews where the company is going with their products and what the main investment areas are to be in the future. Also, the product's focus areas are discussed and analysed. If there is a need to make changes in the focus areas, these inputs are gone through and, based on the discussions and decisions, the future tasks and issues are changed. Thus, the weekly meetings are for implementing the practical changes that need to be made.

Continuous product planning means that product roadmaps are reviewed and updated continuously so that they are in sync with the company's strategy plan, business area plans and release plans. Also, the content of the product roadmap is kept tentative, which helps to reduce external linkages needed for following and implementing the roadmap. Then, for example, changing interest points, dropping something out of the roadmap or dramatically changing the focus areas of the roadmap is easier. The content of the roadmap may even change radically every quarter, so the rolling forecast for the next year is considered to be the most practical way of planning. The product planning is conducted inside the product domain. The product management guides the actual R&D teams work by creating product roadmaps. A product roadmap also relates to the business area strategy, as the strategy is being implemented with the help of technology. The product roadmaps include information relating to the products and their variants and features as well as high-level descriptions of the user stories, and requirements. The product roadmaps are created within the time frame of one year, in which the first half year is a clearer, so-called high-confidence plan, and the second half of

the year involves more tentative content. Product roadmaps are revised on a weekly, biweekly or quarterly review cycle. These review practices vary between products as they are different in nature. For example, product maturity, the legacy and complexity of the product and the customers and sales channels, all affect product planning. Input to the product roadmaps comes from various sources, for example, through the product managers and sales. The product management review all the new feature requests, for example, whether it is a platform, product, or customer specific asset. If the feature is considered to be vital, it will be analysed further; what it means in relation to the schedules and costs, which also requires the involvement of technical staff. Then the technical effect is analysed together with the cost impact, for example, how much effort must be calculated, and how much it affects the product's timetable and all the costs. Also, at the same time, market and customer needs are analysed, how much adding some specific feature would be expected to increase sales, for instance.

Continuous release and feature planning means that release and feature plans are reviewed continuously instead of on a predefined and exact date. However, it does not mean that everything needs to be changed daily, all the time. If a change is needed, a set of features may be fixed. Simply by monitoring progress and recognising that the amount of work remaining matches with the capacity available constitutes continuous planning. Release planning is conducted by the R&D team who create a feature roadmap based on the current product roadmap. The time frame for release planning or feature roadmapping is two months. It involves practices relating to requirement management, such as prioritisation and estimation. Features are typically implemented and verified with two-, four- or six-week release cycles, depending on the team. The team can typically decide the sprint length; three weeks is quite optimal, but weekly releases are suitable in situations with many changes. Also, release plans or feature roadmaps are reviewed within a daily review cycle, because they focus on implementation and verification issues in daily work. As the products are managed by handling features in the product roadmaps, the teams are managed through stories in the feature roadmap. The features are split into stories to be implemented by the team. In addition, a project plan is one of the team's outputs, which is co-created so that everyone can see the same artefact and suggest improvements. The project plan includes a scope, resources and schedule. Continuous project planning by the team is conducted as follows. First, the team makes estimates based on the scope; then they make allocations of resources, and finally, they have a project plan. The main activity of continuous project planning is balancing these three elements, which also serves to establish the priority of the items in the scope. The main working method of the team is 'war-room workshopping', which enables proper communication and less confusion, as well as improving team spirit and enabling visual planning.

Continuous planning through roadmapping is applied to present the hierarchy of several plans together with their time frames. The roadmap structure helps to provide an integrated and shared picture of the whole organisation and its planning activities. Figure 10 presents the empirical findings of the levels of continuous planning and the associated activities through a roadmap structure.

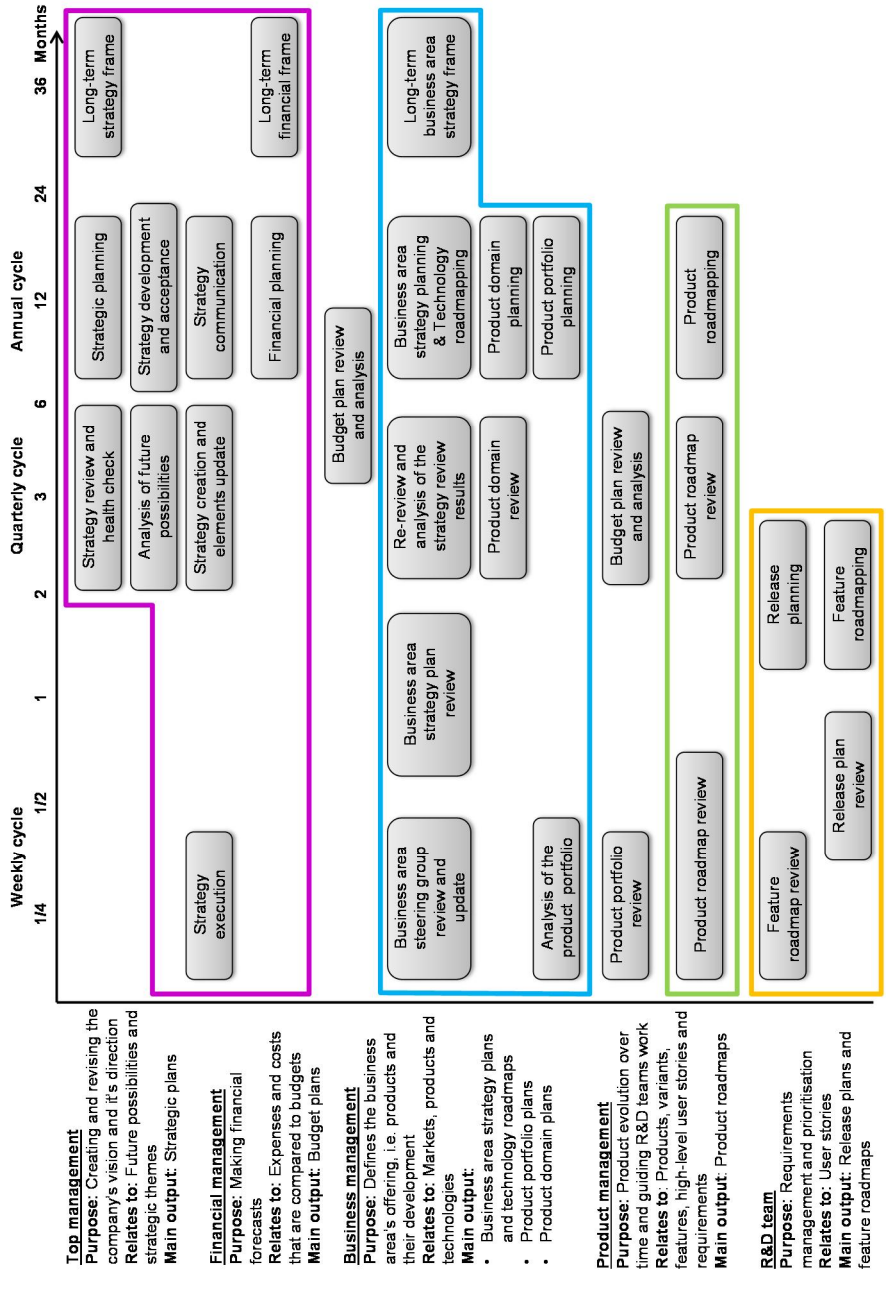


Figure 10. Continuous planning through roadmapping.

In Figure 10 the roadmap structure on *the horizontal axis* describes the roadmapping purpose on the organisational level, which refers to the level of planning together with the main participants and their main activities. It forms a higher-level view of the overall planning process as well as classifying the main entities of planning, which are highlighted with different colours in the figure. The planning activities outside the main entities of planning (highlighted with colour frames) are so-called connecting activities between the levels of planning. *The vertical axis,* in contrast, describes the content emphasis of the roadmap positioning within a time frame, and identifies the main cycles of planning (that is weekly, quarterly, and annual cycles). The activities associated with continuous planning as well as their time frame are revealed as they are presented with the help of the roadmap structure. Furthermore, the roadmap structure not only highlights the masses of planning but also the lack of planning. The roadmap structure brings out the synchronisation between levels of planning as well as pointing out the most important and challenging activities associated with organisational planning. These findings are discussed in more detail and compared to the findings of related work next.

The figure presents the main *levels of planning* for an agile–lean organisation as follows: strategic, financial, business, product, and release planning. These are quite similar to the levels of planning presented in the related work (in Subchapter 2.3): strategic, business and financial, portfolio, product, and release planning. The main *participants* of the planning activities are according to empirical findings respectively: top management, financial management, business management, product management, and the R&D team. The participants presented in the related work (in Subchapter 2.4) are: the executive committee, business and financial management, portfolio management, product management, and the agile team. The biggest difference lies in the middle level of planning, which is called portfolio planning according to the related work, and business planning according to the empirical findings. Based on the related work, this middle level of planning is purely related to portfolio management, whereas based on the empirical results it is much more than that. In addition to portfolio planning it also involves business area strategy planning and product domain planning, but also synchronisation between the levels of planning both up to strategic planning and down to product roadmapping. Thus, it is not only being one of the main levels of planning, but is also one of the most challenging activities to conduct. This is because business planning requires both technical and managerial skills from business management as the business area strategy plans are quite strongly product and technology focused, though they are also responsible for synchronising the plans between product domains and seen that the product domains are progressing according to the company's strategy. Therefore, as a solution, the empirical results point out that business planning is co-created together with business area teams that also include people from product management and development in addition to business management so as to get more knowledge involved in the process.

Similar to business planning, the empirical results also bring out the importance of product management, as they also involve *synchronisation between the*

different levels of planning. Product planning provides information both up to business and strategic planning and down to team-level planning through product roadmaps. Product roadmaps improve the visibility of planning by providing information on how product development is progressing along the company's strategy. Hence, the product roadmap is one of the main outputs of continuous planning throughout the organisation. Therefore, especially product roadmapping, should be done in a continuous manner in order to synchronise the company's strategy, business, and release plans. As with business planning, the product roadmaps are commonly co-created together with product management and development, and are then presented to business management so as to get more perspectives involved in creating and validating the content.

Both business management and product management are also involved in conducting the other so-called connecting activities between the levels of planning, as presented in Figure 10. These *connecting activities* (outside the colour frames) relate to reviewing budget or product portfolio plans. Both business and product management will review and give feedback on the budget plans created by the financial management, thus connecting the financial, business, and product planning together. In addition, product management also reviews the product portfolio plans created by the business management so that both the product and business level plans are in line with each other.

Even though the planning activities (especially with regard to team, financial, and strategic planning) share similarities between the related work and the empirical findings, *the time frames of the planning activities* are different when comparing the results. In the related work (in Subchapter 2.4), the time frame constitutes days to eighteen months and planning is conducted in four main time periods namely: a day to three months, three months to six months, six months to one year, and a year to a year and a half. Instead, based on the empirical findings presented in Figure 10, the time frame constitutes days to 36 months, as a majority of long-term plans looked three years ahead. Thus, it was realised that even though continuous planning is mainly understood as short-term planning, it does not remove the need for long-term planning, as strategic, financial, and business planning are required to steer the company in the right and desired direction both in the short-term and long-term. Furthermore, in the light of the empirical findings, planning is conducted in three main periods through weekly, quarterly, and annual cycles. Commonly, strategic, financial, business, and product plans are created annually, but the practices of how they are reviewed vary. Where strategic plans are reviewed quarterly, budget plans are reviewed bi-annually or quarterly, and business and product plans are reviewed weekly. Instead, the team level plans are created within a time frame of less than two months and reviewed even with a daily planning cycle.

The empirical findings summarised in Figure 10 also bring *out the large amount of planning well as the lack of planning.* It seems that most of the planning activities occur either within the time frame of three months to a one year or with less than two months. The planning relating to strategy and business in particular is typically conducted yearly and the plans are reviewed quarterly. In contrast,

most of the team- and product-level planning are conducted within two months and the plans are even reviewed daily. In addition, the long-term plans relating to strategy, finance, and business are made within a time frame of three years. Instead, the lack of planning relates to the fact that there is no long-term planning related to products or releases. Also, there is not much planning activity relating to strategy or finance with less than a three months' time frame. Even though yearly planning is commonly used, it might not be the most appropriate type. For example, yearly planning is quite inflexible and major changes are quite difficult to make with regard to budgets, for instance. Therefore, it is realised that companies need to be able to react and change plans in the course of the year.

RQ3. How is continuous planning conducted in continuous deployment-driven software development?

There is no formal definition of CD in the current literature; however, there appears to be some level of agreement amongst scholars that CD refers to the ability to bring valuable product features to customers on demand and at will (deployment), in series or patterns with the aim of achieving continuous flow (continuity) and in significantly shorter cycles than traditional lead-times, from a couple of weeks, to days or even hours (speed). Thus, CD builds on three major concepts: 1) deployment, 2) continuity and 3) speed. Especially, in relation to continuous planning, the difference between continuity and speed should be distinguished. Deployment shows intention or the ability to bring a software product or system or service to the production environment in order to be used by the customer. Hence, CD means the ability to bring valuable product features to customers on demand and at will. Continuity emphasises continuous series or patterns so that the software is deployed at a specific time, repeatedly, providing a continuous evolution of software functionality with the aim of achieving continuous flow. Speed focuses on significantly shorter cycles than traditional lead-times, preferably near to real time, at will or on-demand, for example, from a couple of weeks to days or even hours.

CD challenges and changes traditional planning towards continuous planning, whereas traditional software planning tends be performed cyclically and is usually triggered by the annual financial year. In the context of CD, continuous planning is mainly achieved through fast and frequent releases, hence relating to release planning. Fast release is the ability to release software whenever the organisation wishes, based on need, which can be weekly or daily. It is about accelerating the release cycle by shortening the release cadence and turning it into a continuous flow: that is, from months to weeks, or from six months or eight weeks to continuous flow. However, achieving fast release in the form of continuous flow is not free of charge. Shrinking the release cycles down to fortnightly, weekly, semi-weekly and finally at-will, takes months of preparatory work to get the deployment process streamlined and automated. Besides continuous planning, there are also other mechanisms to achieve fast and frequent release, such as automation, close

interaction with customers, having a clear release process, a release management workflow, and a continuous delivery workflow.

However, CD should not be confined to software development and planning alone, and the flow should happen within the overall product development cycle, where software and release planning is just one aspect. Therefore, continuous planning in a CD context includes all the activities from strategic and business planning to product, portfolio, and release planning. In all, the planning activities should be done more frequently so as to ensure alignment between the needs of the business context and software development, as well requiring tighter integration between strategic and business planning and execution. The tighter integration between planning and execution is required in order to achieve a more holistic view in planning throughout the organisation. This is not yet a fully studied area of research, as the related literature focuses mainly on release planning. Further research is therefore required.

How has planning changed for agile and lean software development from roadmapping to continuous planning?

The empirical findings are summarised in Figure 11. The figure shows the levels of planning for an agile–lean organisation and defines their relation to roadmapping. The figure also brings out the cycle of continuous planning and provides information on the planning intensity at each level, together with their main participants referred to in the figure as: planning, analysis, review and update, and change prioritisation.

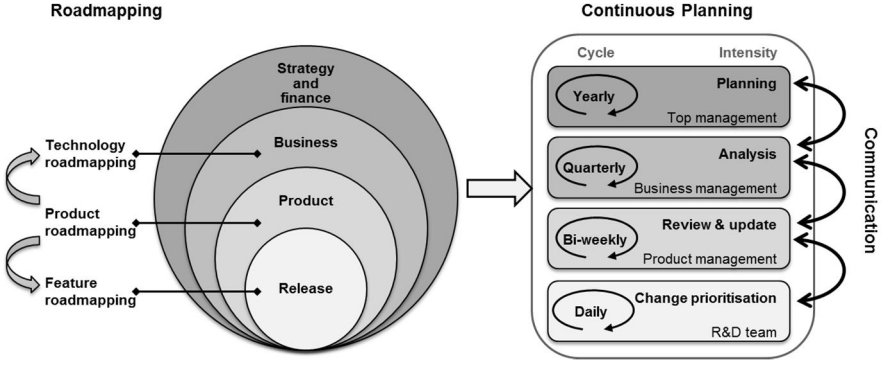


Figure 11. From roadmapping towards continuous planning.

The main levels of continuous planning are as follows: strategy and finance, business, product, and release. Strategic and financial planning is conducted by the top management with a yearly planning cycle. The content is planning-intense, in which the company's strategy is compared to product goals. Business planning is conducted by the business management with a quarterly planning cycle. The

planning is analysis-intense, in which product goals are compared to results. The analysed plans relating to technology or product roadmaps are communicated to product management and those relating to budgets to financial management. Product planning is conducted by the product management with the bi-weekly planning cycle. This planning is review-intensive in nature, in which products are compared with revised products. Release planning is conducted by the R&D team with the daily planning cycle. It focuses on changed prioritisation and involves the actual implementation work and re-prioritisation of features.

Business planning relates to the company's strategy plan being implemented with the help of technology. The main output of business planning is the *technology roadmap*, which defines the company's offering including the products, their development and how they are implemented. However, it is a high level document without direct links to requirements. Requirements are described at a higher level in the product roadmap and thereafter in more detail in the feature roadmap. *The product roadmap* is the main output of product planning. It includes information relating to the products and their variants and features, as well as high-level descriptions of the user stories and requirements. Also, it structures and arranges product development in the order that they are to be done and when and by whom. The R&D teams use product roadmaps to plan their work and to create a feature roadmap. *The feature roadmap* is the main document in the release planning, in which the features are split into stories to be implemented by the team. It involves prioritisation and the change management of features. Even though it seems that strategic and release planning are separated and the people creating these plans are different, there is synchronisation and communication between the levels of planning. In particular, the product roadmap is one of the main documents of organisational planning as it provides information both up and down, since it relates to the company's strategy plan, business area plans, and release plans. Hence, it is used both as a basis for business planning as well as release planning. Product roadmaps help to synchronise information between the levels of planning, but also provide a linkage between product management and development, as well as between product management and business management.

Change in planning practices

In the light of these results, it can be concluded that planning practices have changed in the course of the study. The change has affected both the scope and the schedule of planning.

With regard to the scope, when the research was started, roadmapping was an emerging concept in software development. Back then, in so-called traditional software development, roadmapping related to the development and planning of products with the help of product roadmapping. It was commonly created by a product manager, who was typically the owner of the roadmap, being responsible for collecting input and making the required changes to the roadmap. Thereafter, with agile and lean software development, the planning has moved downwards in

the organisation, closer to the actual implementation and team level planning. The planning practices have changed towards cyclic planning in small and frequent cycles and iterations. They are conducted as release and iteration planning with the help of feature roadmapping, which is commonly created by a product owner who is part of the R&D team. Now, as the software development practices are moving towards continuous deployment with the ability to continuously develop software functionality, continuous planning and transparency of information is highlighted. It has been realised that a wider perspective on planning is required than currently considered in the agile–lean organisation. Continuous planning throughout the organisation is required in order to enable continuous software engineering, including also such areas of roadmapping as strategic, financial and business planning.

With regard to the schedule, the time frame of plans has shortened remarkably in recent years. It has been realised that it is difficult to predict far into the future, and making long-term plans with inaccurate content is useless and is actually against lean principles. Based on empirical findings, none of the companies created long-term plans for five or ten years ahead, as presented in the related literature (e.g. Phaal and Muller 2009). Currently, the long-term strategic plans are created for two or three years ahead. Also, the time frame of the short-term plans such as product roadmaps has shortened radically during the course of the study. When the research was started, it was common that product roadmaps were created for three years (e.g. Lehtola et al. 2005). Similar results were received through the empirical study (that is initial inquiry) showing that product roadmaps were created for two to three years ahead. Currently, with the adoption of agile and lean software development practices, the time frame of product plans has shortened to a one-year view, in which the first half of the year is more accurate and the rest of the year involves more tentative content. Some of the case companies even clarified that making product plans for one year ahead is difficult and some of them even pointed out that they would rather plan only for half a year ahead.

In light of the empirical results, strategic and financial planning are still considered rather project-based efforts, as they are done annually, instead of the ideal continuous planning with planning purely based on need. Even though it is recognised in the related work that some literature supporting practices of continuous strategy or rolling budgeting exists (e.g. Koenigsaecker 2009, Rickards and Ritsert 2012), it seems that it is not yet applied among the companies. However, the companies have realised that strategic planning should be done more often, and some companies have even adopted a continuous review process while others are still considering adopting a quarterly review process. Similarly, the results reveal that one year is too long a time frame to make budget plans. A half year would be a more accurate time frame, but it is realised that biannual budgeting can be really time-consuming, since it takes lot of effort to get it done and approved. Therefore, creating a yearly budgeting frame with a rolling review process might be more convenient than bi-annual budgeting. The team-level planning is quite fast-paced already, as it is in the agile–lean organisation, where it

is conducted with less than a two-month time frame and with less than a week's review cycle. Actually, the empirical results point out that there is no need to make the team level planning any faster, but instead the goal is to reach a freer release cadence so that the team can decide the release and sprint length by themselves.

Reasons for the change in planning practices

The background of continuous planning lies in agile and lean software development practices. A recent evolutionary step from agile and lean software development is continuous deployment, which refers to the organisational capability to develop, release and learn from software in rapid parallel cycles, such as hours, days or a very few weeks. This phenomenon is a logical progression of extending agile and lean approaches, where the step between development and deployment is minimalized in order to deploy code immediately to the production environment for customers to use. CD extends agile and lean practices by moving from cyclic to continuous value delivery. This evolution requires not only agile processes at the team level, but also integration of the complete R&D organisation, parallelisation and automation of processes that are sequential in the value chain and constant customer feedback. Where agile and lean practices focus on speeding up the development process at the team level (with such methods as Scrum), CD moves beyond the concept of agile and lean towards a situation in which software functionality is continuously deployed to the final customers. Therefore, continuous deployment goes beyond agile and lean software development; thus, these methods and practices are the first steps the organisation can take toward CD and hence are considered as an enabler for CD. Also, CD scales agile and lean practices throughout the whole organisation instead of focusing only on team-level activities. In other words, CD transforms the traditional agile and lean practices and methods into a continuous flow.

The reasons for the change in planning towards continuous planning can be explained by the external and internal challenges that companies face in today's volatile market environments. Both the unstable and turbulent business environment and fast technology development have meant that the future is really difficult to predict, which means that continuous planning is needed. Furthermore, the need for continuous planning arises as the organisations have problems in developing long-term plans due to constant changes in their customer and market bases, as well as in product and technology development. Also, recent financial crises have caused companies to rethink their approaches to planning and to realise the importance of continuous planning from both an operational and financial perspective. Various and sudden changes in the business environment have forced companies to adopt continuous strategic and financial planning practices. For example, large companies and important partners going bankrupt or suddenly disappearing from the market area have caused companies to quickly get new partners and redirect their product offering to these new customers. Therefore, a strategy plan, for example, needs to be changed and redirected a lot in the course of the year. Instead, the internal changes relate for example to the

adoption of agile and lean development practices, which force companies to shorten their product planning and review cycles to months and to shorten team-level planning to weeks or days. While going further towards continuous software engineering, internal changes are required to enable continuous deployment, such as the automation of the delivery pipeline from build and testing to deployment and monitoring.

The reasons for the change can also be explained through the benefits that continuous planning offers. From the strategic perspective, the management is able to show staff the current status more clearly, for example, what they have done and where they now want to be both in the mid-term time frame and in the long-term time frame. Since the strategy is reviewed, followed up, communicated and published quarterly, it leads to increased trust in management, as well as to improved competitiveness and decreased fixed costs. From the financial perspective, the main benefits of continuous planning are that the estimates become more accurate. From the portfolio-level perspective, especially if the team is using a visual Kanban board or wall, a key benefit is that the people in charge of the portfolio level can come to the war-room and actually see what the team is currently doing. The visual wall also makes it much easier to explain why things have been done in a certain way, instead of trying to explain it using backlog excel sheets. From the product management perspective, continuous planning enables a better level of communication, as it brings more people into the planning process. Furthermore, from the R&D project management perspective, a key benefit of continuous planning is that it reduces work-in-progress as well as improving quality levels. The team improves in writing features, since the project performance time becomes much clearer. Thus, the feature level plan becomes more reliable given that there are not so many modifications during the implementation.

The reasons for change can be associated with the benefits achieved while conducting continuous planning in CD-driven software development. These benefits include shorter time-to-market; instant feedback, especially from customers when using proper monitoring and experimentation systems; improved release reliability, partially as a result of a narrower test focus; and improved customer satisfaction and developer productivity. All in all, the most immediate benefit of applying CD is a shorter time-to-market through fast and frequent releases, which is achieved through continuous planning. These benefits are received when the companies shorten their delivery cycles from months or weeks to continuous flow or daily deliveries. Shorter release cycles enable companies to constantly develop, learn and improve their offering based on instant customer feedback and thus, companies can quickly learn what customers value and focus on deploying relevant functionalities that meet customers' expectations. Shorter release cycles enable faster feedback about new features and bug fixes, which makes release planning slightly easier. CD has also been found to increase customer satisfaction and to enable continuous customer feedback. CD allows continual product enhancement and immediate access to new features and bug fixes, which increase customer satisfaction. By using continuous planning at the

release-level, users can adopt new versions of the product faster, bugs are fixed faster and users do not experience significantly more post-release bugs in comparison with traditional release planning. In addition, customers have a chance to evaluate the enhancements and provide feedback immediately and in a continuous way (that is continuous customer feedback), which improves communication between the company and its customers. Furthermore, closer interaction with customers enables enterprises to monitor and collect instant field data on their customers and software's behaviour. The main advantage is that companies have the chance to rapidly sense, understand and improve their offering based on actionable metrics and data. Closer relationships with customers further facilitate rapid innovation. Continuous and instant customer feedback allows companies to invest their resources in developing relevant functionalities and innovation initiatives. Faster feedback can also mean cheaper development, since the R&D organisation can then spend time developing the right things rather than correcting mistakes in functionality, which is not necessarily what the customer wants.

Effects of the change in planning practices

The change towards continuous planning has various effects. In order to achieve continuous planning, organisations need to be capable of changing their operations and adapting their mind-set towards continuous planning and transparency throughout the whole organisation. Organisations need continuous visibility of their development and operations in order to provide information to all employees. On the one hand, operational transparency is related to increasing the visibility of the performed work, to planning and executing actions in relation to the fulfilment of one set of defined criteria. On the other hand, development transparency is related to identifying the potential technological and cultural barriers to implementing increased transparency and improving learning in connection to needs that arise due to increased transparency. Also, knowledge sharing has become increasingly important inside the organisations, as both push (e.g. training courses) and pull type (e.g. publishing information via wikis and articles) of knowledge sharing is needed. Furthermore, beyond this visibility, continuous competency planning and development are needed. Employees' competencies should be able to adapt constantly and change through continuous analysis, development activities and evaluations as to the successfulness of actions.

Continuous planning also has other challenging effects that relate both to people, whether they are, for instance, willing to be engaged and involved, and to transparency, how to make goals visible and suit a continuous setting, for instance. People may view continuous planning as losing authority, and thus, for example, the culture towards continuous planning needs to be embraced throughout the organisation. The main expectations of continuous planning are related to turning the goals into a continuous planning mode and how to fulfil long-term issues. When the longer-term goals are taken into account, they should be

quite quickly optimised into shorter-term business productivity. Therefore, companies need to consider whether they are sufficiently taking the longer-term architecture and business strategies into account. But all in all, continuous planning does not remove the need for long-term planning. Instead, it is realised that company management needs to consider the long-term perspective of various issues at the same time as the background, such as resourcing and competency needs, as well as a location and technology strategy. Yet not all of these actives might be visible to the team. Visibility could be improved by using a tool that would present information relating to various levels of planning. In addition, one of the main obstacles to continuous planning is that many parts of the organisation would rather work with longer-term plans than three months. For example, the business areas prefer to have a view from six months to one year, but in order to conduct continuous planning, the teams might be only promised a three-month view, while the longer view is only an outlook. The short-term planning cycle requires a major mental shift for many actors, especially the outside R&D team, and the capability of doing the work in a continuous manner must be visible all the time to the rest of the company.

Furthermore, moving towards CD implies challenges for the whole organisation, such as: customers' unwillingness to receive continuous product updates, increased quality assurance (QA) efforts, and difficulties applying CD in the embedded domain. Similar to continuous planning, moving towards CD is an evolutionary process and requires investment in deployment processes, as well as changes in people's mind-set and organisations' way of working. Therefore, in the context of continuous software engineering, where organisations are required to develop, deliver and learn in fast and parallel cycles, it is profoundly important to establish an agile thinking culture among individuals, teams, as well as upper management levels. Furthermore, even though customers seem to be more satisfied, customer unwillingness to accept CD is another challenge. For example, customers are reluctant to accept new functionalities mainly because of the poor quality of releases. Similarly, while rapid release has numerous benefits and strongly supports shorter release length, at the same time it increases the test efforts. This stems from the fact that more specialized testers are required to sustain the testing effort in a rapid release model. In addition, CD requires establishing an effective QA process and new mechanisms to ensure backward compatibility of enhancements. In addition, further difficulties exist in release planning and managing the roadmap in a fast-paced environment and risks associated with gathering user feedback from a limited population that may constrain the software's evolution or even mislead product development.

5.2 Evaluation of the results

In this subchapter, the research results are evaluated. Firstly, theoretical contributions and the empirical implications of the research are discussed. Then,

the reliability and validity of research are presented, and finally, the limitations of the results are discussed.

Theoretical contributions

In this study, a literature review was conducted comprising the theories of agile and lean software development (see Subchapter 2.1), roadmapping (see Chapter 2.2), and continuous planning (see Subchapter 2.3). Based on this literature review, a figure describing the current planning practices through roadmap structure in the agile and lean software development context is presented (see Subchapter 2.4). The implications of the key findings are evaluated with respect to these main theories.

With regard to *the current literature of agile and lean software development*, Cohn (2006) has presented a planning onion consisting of the following levels of planning: strategy, portfolio, product, release, iteration, and day. In this related literature, the responsibility for planning is moved to agile teams, and thus planning mainly relates to iterations or releases. The other levels of planning outside the concern of agile teams (that is strategy, portfolio, product) are not defined with the same level of detail. Hence, this thesis points out that the agile team level planning is necessary but not sufficient and a wider perspective on organisation planning in the agile and lean software development context is required. Thus, this thesis collects all the empirical findings together and presents the results with the same level of detail in all the following levels of planning: strategy, finance, business, product, and release.

With regard to *the current roadmapping literature*, the current empirical practices related to roadmapping are identified and presented in the thesis. These practices focus especially on product roadmapping, as it has been seen as one of main activities in software development. Furthermore, this thesis improves the current understanding of roadmapping by defining the roadmapping process along with the main participants, as well as the main elements of the roadmap structure (i.e. the layers of knowledge). The roadmap structure not only enables the company to present the same level of detail to the planning (i.e. the levels) but also to go further with the planning onion theory and beyond the levels of planning by revealing also the associated planning activities, as well as the time frames and cycles of planning. Furthermore, even though it has been pointed out in the current literature (Phaal et al. 2004b, Groenveld 2007) that the roadmapping process is different between companies, and that there is not just one process model to be adopted, this thesis demonstrates something different among Finnish and Swedish ICT companies. Based on the empirical results, the case companies had not only similar levels of planning and time frames, but also similar participants conducting the planning.

The current literature on continuous planning lacks a definition and a more profound understanding of the way continuous planning is conducted throughout the organisation in the agile and lean software development. This is recognised not only through the related literature (e.g. Fitzgerald and Stol 2014) but also

through empirical findings presented in the related original publications. Accordingly, it is evident that continuous planning requires a wider perspective than is currently considered. It should be examined from a broader, even more continuous perspective than currently exists. It is realised that continuous planning is not only a team-level activity, but rather it involves higher-level planning as well, for example, strategic and financial planning. In the related literature, Fitzgerald and Stol (2014), for example, emphasise continuously assessing and improving the link between business strategy and software development, although they do not define how such a link could be achieved or how a business strategy should be done in a continuous manner in order keep it in line with the continuous software development. Hence, this thesis addressed this research gap by increasing knowledge of continuous planning in an agile–lean organisation. Furthermore, as a unique contribution to the field of research, this thesis views *continuous planning through roadmapping*. The roadmap structure brings out the main levels of planning together with their time frames and the participants involved. Thus, this thesis defines how continuous planning is conducted at various levels of the agile–lean organisation (that is strategy, finance, business, product, and release) as well as providing information on the planning intensity at each level and identifying the main participants of that specific level of planning. The roadmap structure also reveals the masses and lack of planning and reveals the synchronisation between levels of planning, as well as points out the most important and challenging activities associated with organisational planning, which is currently missing in the related literatures.

The main theoretical contribution of this thesis is revealing *the change in the planning for agile and lean software development.* It has been realised that the current planning practices used in software development, such as product roadmapping, are evolving towards continuous planning. Unique to this field of research, this thesis reveals the reasons for this change and points out what kind of challenges these new planning practices cause and what the associated benefits are. Additionally, this research sheds light on the new and not yet fully studied area of continuous deployment, which is believed to be the next evolutionary step in agile and lean software development (e.g. Olsson et al. 2012). This new research area is tackled with a wide systematic mapping study. Through the lens of the systematic mapping study, is it recognised that continuous planning is indeed part of the emerging new trend of continuous deployment and further it is seen as one of the mechanisms to achieve continuous deployment. The thesis provides further insights into the phenomenon of continuous planning and development, which all in all enriches the current body of knowledge as well as provides material for further research.

Empirical implications

This thesis provided results on the practical implications received through case studies. The empirical implications relate to the recognition of change in the planning practices among software development companies from roadmapping towards

continuous planning. Practical evidence is provided from both of these main areas of research, namely roadmapping and continuous planning. In what follows, these main aspects of the practical implications from this thesis are discussed.

The findings related to roadmapping help companies to improve their own roadmapping practices by focusing their improvement efforts on the most critical parts of the product roadmapping process, that is capturing and prioritising features, and changing the management of the roadmap. The findings also provide empirical understanding and experiences of the most important activities in collaborative product roadmapping and bring out the challenges that inter-company collaboration sets for the roadmapping process.

The findings related to continuous planning provide actual practices in how to conduct continuous planning at the various levels of the organisation, such as strategic, financial, business, product, and release planning. These practices are the ones that companies can accommodate while seeking to develop or improve their current planning processes and practices towards continuous planning. The empirical implications also show how to use continuous planning through roadmapping in the agile and lean software development context. The roadmap structure provides information for the various levels of planning together with the main roles and planning activities. Furthermore, it brings out the main activities associated with planning, but also highlights the bottle necks as well as lack of planning.

Most importantly this research uncovers practical implications for software development organisations that have recognised the need for continuous planning or have already changed their planning practices towards continuous planning. One of the main implications for practice stem from the reported reasons for why change in planning has occurred, and what the main consequences of the change are. These issues are vital for practitioners to consider while changing their current planning or development practices. The findings also bring out other aspects related to continuous planning, such as leadership, transparency and competency development, which companies can consider while improving their current planning practices. Also, the findings of this research help industrial companies to better prepare for the internal and external challenges arising both from going beyond agile and lean software development towards continuous deployment and from the constant changes and surprising events in the business environment.

Reliability and validity of the research

This thesis builds on five original publications. Together, these publications form a consistent body of knowledge bringing out the change in planning that the software development industry is currently facing. The papers were published over the course of a relatively long time frame, almost ten years, but this is reasonable in order to describe the phenomenon of change. All the results presented in these publications have been reviewed, evaluated and then published in high-quality scientific forums; that is highly rated journals and conferences of repute. Also, the reliability and validity issues are separately discussed in the original publications.

The research results presented in the original publications are based on empirical data. Papers I and II present research results from the initial inquiry and Papers III and IV are based on a multiple-case study involving three case companies. The data in Papers I–IV were gathered through various sources such as a questionnaire study, interviews, a series of meetings and workshops, and through the analysis of company-specific internal memos and material. All these empirical data were gathered through real industrial settings and the data together with the results were reviewed and validated during the research process by the case company representatives. The research results presented in Paper V are based on a systematic mapping study. In order to provide reliable results, all non-peer-reviewed scientific studies, books, book chapters, and short papers were excluded from the study.

In order to establish the reliability of this research, triangulation of the study's qualitative data was performed as a validity procedure to increase the precision of the empirical research. The purpose of triangulation is to approach a studied object from various angles to provide an in-depth understanding of it (Runeson and Höst 2009). Accordingly, four different types of triangulation may be used: data (or so-called source) triangulation, observer triangulation, methodological triangulation, and theory triangulation. *The data triangulation* of this research project served to strengthen its research results, as more than one data source was used (that is interviews, meeting and workshop as well as company-specific material) and the same kind of data were collected on different occasions. The results of this research were also strengthened by *observer triangulation*, as there was more than one researcher involved in collecting the data, and the representatives of the case companies played various different roles. *The methodological triangulation* was strengthened by the fact that different types of data collection methods, such as qualitative and quantitative methods, were combined during the research, especially in the initial inquiry part of the research. The research results are also bolstered via the *triangulation of theory* in this research, as alternative theories or viewpoints were discussed in the presentation of the research results.

In order to establish the validity of this research, commonly used criteria to evaluate the validity of the empirical research (Yin 2003, Easterbrook et al. 2008) were used. Four tests can be used to establish the quality of the research including construct validity, internal validity, external validity, and reliability.

Construct validity focuses on whether the theoretical constructs are interpreted and measured correctly. It reflects to what extent the measures studied represent the intentions of the research and if what is being studied is according to research questions. There is a threat to construct validity when the measured variables do not correspond to the intended meanings of the theoretical terms. For example, problems occur when the constructs discussed in the interview questions are not interpreted in the same way by the researcher and the interviewees. There are three tactics to increase construct validity (Yin 2003): using multiple sources of evidence, establishing a chain of evidence, and having a draft case study report reviewed by key informants. The research covered three case companies and

several other companies involved in the initial inquiry, thus having multiple sources of evidence. Also, all the interview reports and case descriptions were reviewed by the company representatives.

Internal validity focuses on the study design and particularly on whether the results really do follow from the data. It is a criterion related to causal relationships. For example, if one is studying whether a certain factor affects the factor being studied, there is a risk that a third factor also affects the one under study. There is a threat to internal validity if the researcher is not aware of this factor and does not know to what extent this third factor affects the one that is investigated. However, this logic is not applicable to descriptive or exploratory studies that are not concerning with making causal statements, as is the situation in this research. Furthermore, the research is conducted in a real-life setting instead of a laboratory environment, and hence the factors affecting the results cannot be isolated. As the research goal was to understand and discover the current practices of roadmapping and continuous planning in practice, the real life industrial projects were seen as necessary. The internal validity has been supported in that the results have been analysed and validated by several people: the author, co-authors of the original publications, company representative, and the case participants.

External validity focuses on whether claims for the generalisability of the results are justified. This depends on the nature of the sampling used in a study. In case studies, the intention is to extend the results to similar cases in which the findings would be relevant. Thus, during the analysis of external validity, the researcher tries to analyse to what extent the findings are of relevance for other cases. In this thesis, this is supported by using a multiple-case study approach to reveal similar topics emerging from several companies.

Reliability focuses on whether the study would yield the same results if it were to be conducted all over again by other researchers. Thus, it considers the extent to which the data and the analysis are dependent on the researcher. There is a threat to the reliability of the study if the case study research procedure is poorly documented. Without proper documentation, the earlier case study is difficult to repeat. Therefore, all the vital information related to conducting this research are carefully documented, including the research, interview and questionnaire questions, as well as storing all case material. Furthermore, the NVivo empirical data analysing tool was used to store, analyse and classify the empirical data. Also, the case studies included in this research were reviewed and analysed by other researchers than the author, providing evidence of reliability.

Furthermore, in relation to case studies, Easterbrook et al. (2008) identify that one major weakness of case studies is that the data collection and analysis are more open to interpretation and researcher bias. For this reason, an explicit framework is needed for selecting cases and collecting data. Even though an individual case study often reveals deep insights, the validity of the results depends on a broader framework of empirical induction (Easterbrook et al. 2008). Hence, it was realised that the use of multiple case studies would offer greater validity to the research. Thereafter, the case companies were carefully selected.

All the companies involved in this research are operating in the field of ICT product and service development and are large companies with more than 1,000 employees. Also, all the case companies had transformed their organisational and software development practices to use agile methods and a lean development approach. Therefore, all the case companies involved in the multiple-case study were believed to fit well to the overall research context.

Easterbrook et al. (2008) also point out that all research conducted in industrial settings brings a number of challenges. It can be very hard to gather data to ascertain what practitioners actually do or what needs to be improved in the organisation, rather than what practitioners say they do or what they think requires improvement. In return for access to the organisation, the researcher usually has to give up some control. For example, it is difficult to observe and document findings without interfering with the situation observed, especially when the industrial partners want to know in advance what the expected outcomes are. It is often difficult to know whether changes are made through involvement in the research or whether they would have occurred anyway. Finally, obtaining permission to publish the results can be a challenge. Delays in publication are likely if the organisation has concerns about the inclusion of confidential data or insights in the research.

Limitations of the results

A common argument regarding the case study research method is that generalisation of its results is limited (Yin 2003). In general, generalisability refers to the degree to which the original data are a representative of a larger population, thus a multiple-case study research method was used to gather data from various companies and sources. But, due to the fact that only three industrial companies were analysed in more detail (referring to the multiple-case study), it cannot be said that the results can be applied to industrial companies in general. Furthermore, the results of this study are weakened due to the fact that almost all the case companies involved in this research were Finnish companies, with the exceptions of one of the case companies in the initial inquiry and one of the case companies in the multiple-case study were Swedish companies. The research results might have been different if the case companies would also have included other European, Asian, or American companies. Thus, the research results cannot be generalised to a larger context involving other countries and cultures, as the research has been done in one country mainly. In addition, the results of this study can only be applied to a certain extent to smaller companies, as all the multiple-case study companies were considered to be large. Therefore, further research will be needed in this field; not only to involve companies with different geographical and cultural backgrounds, but also of various sizes.

The empirical data are mainly based on qualitative interviews. Thus, it is possible that all the planning practices together with their impediments and enablers have not been mentioned in the interviews and might play an important role regarding the research results. The interviewees also possess subjective

perspectives as to the questions asked concerning how the continuous planning process is viewed and defined within each case company. In addition, the interviewees' answers do not speak for the collective opinions of others in the case companies. Furthermore, the limitations of the results are weakened due to the fact that there were only three interviews and interviewees in Case C. Also, all of the interviewees were from the same business area and none of them participated to the company's strategic planning.

5.3 Conclusions and future research

This thesis summarised five original publications in the field of research. The work presented in this thesis contributes to the understanding of change in the planning for agile and lean software development from roadmapping towards continuous planning. This thesis provides empirical evidence on these two main topical areas of the research, roadmapping and continuous planning, in the context of software development.

This thesis defines a roadmap as a plan or a leading map of the company's future directions. Roadmapping, in contrast, is the process of creating and revising roadmaps. Continuous planning is about implementing planning and roadmapping practices continuously, so that plans are created and revised based on need, instead of having predefined and regular planning occasions. The roadmap structure consists of the levels of planning and the time frames of the plans. The roadmap structure is applied to gaining an integrated and shared picture of the whole organisation and its planning levels and activities. The horizontal axis describes the roadmapping purpose on the level of planning, revealing also the main participants and their main activities at each level of planning. The vertical axis describes the content emphasis of the roadmap positioning within a time frame, and identifies the main planning cycles. The levels of continuous planning for the agile–lean organisation are defined in the thesis as follows: strategic planning, financial planning, business planning, product planning, and release planning. The main cycles of planning are conducted weekly, quarterly and annually. More longer-term plans are created for three years ahead. The results of this thesis focus on product planning, as it realised that roadmap-based planning in a software development context focuses on product roadmapping. Product roadmaps improve visibility in planning by providing information on how product development is progressing in relation to the company's strategy. Thus, a product roadmap provides information both upwards to business and strategic planning and downwards to team level planning. Hence, the product roadmap provides a link between technology and feature roadmaps, but also synchronises the information between these levels of planning.

Based on the findings of this thesis, it was realised that the planning practices have changed both in regards to the scope and schedule of planning. Planning in agile and lean software development is not restricted to release planning only; instead planning is viewed from a wider perspective involving also strategic and

financial planning. Also, the time frame of the plans has shortened remarkably in recent years from years to months, weeks, and days. The reasons for these changes are both internal and external. Both the unstable and turbulent business environment and the rapid development of technology and new product development practices and shorter product development cycles are drivers for the change in planning. Also, the recent financial crises and constant changes in the business environment pave the way for adopting continuous practices. As a result, it is also revealed that agile and lean software development companies broadly speaking are not conducting continuous planning throughout the organisation. However, it is believed this will change in the near future as it is realised that software development companies are going beyond agile and lean software development towards practices of continuous deployment and planning.

Based on the discussions in these research results, several suggestions for future research on this and related topics can be made as follows:

Firstly, the empirical evidence could be extended by conducting more interviews, especially in Case C. But, also more case companies could be involved in the research in order to gain a more profound and in-depth understanding of the research phenomenon at hand. Also, other research methods, besides interviews and workshops, could be used to gather data. For example, using observation could reveal interesting insights into the planning occasions themselves that would not have been researched otherwise. The existence of broader case data would allow for comparing and contrasting the findings of different cases. As this research focused on large organisations, it would be interesting also to collect data from small and medium-sized companies in order to compare the results between the different sized companies. In particular, there were indications that small companies would be facing the same challenges as large companies even though the organisational structures are much thinner and plans are done with a much smaller group of people. Thus, it would be interesting to discover whether the problems in planning are caused by human-related issues such as poor communication and personal characteristics, or by the planning practices and processes themselves.

Secondly, as the majority of the case companies did not utilise the practices of continuous planning throughout the organisation, it would be interesting to find out what kind of problems would be encountered if they did. Also, it would be interesting to ascertain what kind of tools would be needed to support continuous planning throughout the organisation in order to improve transparency and real-time and relevant data. Moreover, the research could be continued by defining more profound strategic, financial, and business planning practices supporting also continuous deployment to be used especially among software development companies. This work could be continued by identifying the right time of planning so that the continuous strategic planning would not interfere too often in product and team level planning or vice versa. Furthermore, after a couple of years, when companies have adopted the practices of continuous deployment, it would be interesting to see how this has affected its organisational structures and planning practices.

References

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J., 2002. Agile software development methods: Review and analysis. VTT Publications 478, VTT, Espoo, pp. 107.

Agile Manifesto, Manifesto for Agile Software Development. Available: http://agilemanifesto.org/ [Referenced 27.06.2016].

Albright, R.E., 2002. The Process: How to Use Roadmapping for Global Platform Products. PDMA Visions, vol. 26, no. 4, pp. 19-23.

Albright, R.E., 2003. A Unifying Architecture For Roadmaps Frames a Value Scorecard. Proceedings of the IEEE International Engineering Management Conference, pp. 383-386.

Albright, R.E. and Kappel, T.A., 2003. Roadmapping in the Corporation. Research Technology Management, vol. 46, no. 2, pp. 31-40.

Alreck, P.L. and Settle, R.B., 1995. The Survey Research Handbook. 2nd ed. Irwin Professional Publishing, Chicago, 450 p.

Baskerville, R.L., Mathiassen, L. and Pries-Heje, J., 2005. Agility in fours: IT diffusion, IT infrastructures, IT development and business, in: Baskerville, R.L., Mathiassen, L., Pries-Heje, J. and Degross, J.I. (Eds.), Business Agility and Information Technology Diffusion. Springer, New York, pp. 3–10.

Beeton, D., Phaal, R. and Probert, D., 2008. Exploratory roadmapping for foresight. International Journal of Technology Intelligence and Planning, vol. 4, no. 4, pp. 398-412.

Bekkers, W., van de Weerd, I., Spruit, M. and Brinkkemper, S., 2010. A Framework for Process Improvement in Software Product Management. Proceedings of the 17th European systems and software process improvement and innovation conference (EuroSPI), Communications in Computer and Information Science, vol. 99, pp. 1-12.

Bellomo, S., Nord, R.L. and Ozkaya, I., 2013. A study of enabling factors for rapid fielding combined practices to balance speed and stability. Proceedings of the 35th International Conference on Software Engineering (ICSE), pp. 982-991.

Bogsnes, B., 2008. Implementing beyond budgeting: unlocking the performance potential. John Wiley & Sons, Hoboken, New Jersey, 216 p.

Bryson, J.M., 2011. Strategic planning for public and nonprofit organizations: A guide to strengthening and sustaining organizational achievement. Jossey-Bass, San Francisco, California, 325 p.

Butler, D., 2012. Business planning: a guide to business start-up. Butterworth-Heinemann, Oxford, 288 p.

Carvalho, M.M., Fleury, A. and Lopes, A.P., 2013. An overview of the literature on technology roadmapping (TRM): Contributions and trends. Technological Forecasting and Social Change, vol. 80, no. 7, pp. 1418-1437.

Charette, R.N., 2003. Challenging the fundamental notions of software development. Agile Product Management and Software Engineering Excellence, Cutter Consortium, Executive Report, Arlington, Massachusetts, pp. 1-38.

Chong, J., 2005. Social Behaviors on XP and non-XP teams: A Comparative Study. Proceedings of the Agile Development Conference (ADC), pp. 39-48.

Cohn, M., 2006. Agile Estimation and Planning. Prentice Hall, New Jersey, 330 p.

Conn, S.S., 2004. A New Teaching Paradigm in Information Systems Education: An Investigation and Report on the Origins, Significance, and Efficacy of the Agile Development Movement. Information Systems Education Journal, vol. 2, no. 15, pp. 3–18.

Cosner, R.R., Hynds, E.J., Fusfeld, A.R., Loweth, C.V., Scouten, C. and Albright, R., 2007. Integrating roadmapping into technical planning. Research-Technology Management, vol. 50, no. 6, pp. 31-48.

Creswell, J.W., 2003. Research Design: Qualitative, Quantitative, and Mixed Method Approaches. 2nd ed. Sage Publications, Thousand Oaks, California, 246 p.

Darke, P., Shanks, G. and Broadbent, M., 1998. Successfully Completing Case Study Research: Combining Rigour, Relevance and Pragmatism. Information Systems Journal, vol. 8, no. 4, pp. 273-289.

DeGregorio, G., 2000. Technology Management via a Set of Dynamically Linked Roadmaps. Proceedings of the 2000 IEEE Conference, pp. 184-190.

Denzin, N.K. and Lincoln, Y.S., 2000. Handbook of Qualitative Research. 2nd ed. Sage Publications, Thousand Oaks, California, 1065 p.

Dittrich, Y., Pries-Heje, J. and Hjort-Madsen, K., 2005. How to Make Government Agile to Cope with Organizational Change. Proceedings of the IFIP TC8 WG 8.6 International Working Conference, pp. 333-351.

Dove, R., 2005. Agile Enterprise Cornerstones: Knowledge, Values, and Response Ability, in: Baskerville, R.L., Mathiassen, L., Pries-Heje, J. and Degross, J.I. (Eds.), Business Agility and Information Technology Diffusion. Springer, New York, pp. 313-330.

Easterbrook, S., Singer, J., Storey, M.-. and Damian, D., 2008. Selecting empirical methods for software engineering research, in: Shull, F., Singer, J. and Sjøberg, D.I.K. (Eds.), Guide to advanced empirical software engineering. Springer, London, pp. 285-311.

Eppler, M.J. and Platts, K.W., 2009. Visual strategizing: The systematic use of visualization in the strategic-planning process. Long range planning, vol. 42, no. 1, pp. 42-74.

Eriksson, P. and Kovalainen, A., 2008. Qualitative methods in business research. Sage Publications, London, 337 p.

Evans, N.D., 2002. Business Agility: Strategies for Gaining Competitive Advantage Through Mobile Business Solutions. Prentice Hall, New Jersey, 247 p.

Fitzgerald, B. and Stol, K., 2015. Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, pp. 1-14.

Fitzgerald, B. and Stol, K., 2014. Continuous software engineering and beyond: trends and challenges. Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, pp. 1-9.

Groenveld, P., 2007. Roadmapping integrates business and technology. Research-Technology Management, vol. 50, no. 6, pp. 49-58.

Heikkilä, V., Paasivaara, M., Lassenius, C. and Engblom, C., 2013. Continuous release planning in a large-scale Scrum development organization at Ericsson. Proceedings of the Agile Processes in Software Engineering and Extreme Programming Conference (XP), pp. 195–209.

Heikkilä, V., Rautiainen, K. and Jansen, S., 2010. A revelatory case study on scaling agile release planning. Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 289-296.

Highsmith, J., 2002a. Agile software development ecosystems. Addison-Wesley, Boston, 404 p.

Highsmith, J., 2002b. What is Agile Software Development? STSC CrossTalk, The Journal of Defence Software Engineering, no. October, pp. 4-9.

Holmqvist, M. and Pessi, K., 2005. Agility through implementation. A case from a global supply chain, in: Baskerville, R.L., Mathiassen, L., Pries-Heje, J. and Degross, J.I. (Eds.), Business Agility and Information Technology Diffusion. Springer, New York, pp. 173–183.

Hope, J. and Fraser, R., 2003. Beyond budgeting: how managers can break free from the annual performance trap. Harvard Business School Press, Boston, Massachusetts, 232 p.

Ivachtchouk, N., 2004. Agile Software Development. TR-CS-2005-01, Contemporary Challenges in Information Technology. Results of the MSc-Seminars in SS2004 and WS 2004/05, University of Applied Sciences, Department of Computer Science, Augsburg, pp. 86-96.

Jantunen, S. and Smolander, K., 2006. Challenges of Knowledge and Collaboration in Roadmapping. Proceedings of the International Workshop on Software Product Management (IWSPM'06- RE'06), pp. 19-26.

Järvinen, J., Huomo, T., Mikkonen, T. and Tyrväinen, P., 2014. From Agile Software Development to Mercury Business. Proceedings of the 5th International Conference (ICSOB), pp. 58-71.

Järvinen, P., 2001. On Research Methods. Opinpajan kirja, Tampere, Finland, 190 p.

Järvinen, P., 2012. On Research Methods. Opinpajan Kirja, Tampere, Finland, 207 p.

Kameoka, A., Kuwahara, T. and Li, M., 2003. Integrated Strategy Development: An Integrated Roadmapping Approach. Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET), pp. 370-379.

Kappel, T.A., 2001. Perspectives on Roadmaps: How Organisations Talk about the Future. Journal of Product Innovation Management, vol. 18, no. 1, pp. 39-50.

Kettunen, P., 2009. Adopting key lessons from agile manufacturing to agile software product development—A comparative study. Technovation, vol. 29, no. 6, pp. 408-422.

Kitchenham, B. and Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Technical report EBSE 2007-001, Keele University and Durham University, United Kingdom, pp. 57.

Kittlaus, H.B. and Clough, P.N., 2009. Software Product Management and Pricing: Key Success Factors for Software Organizations. Springer-Verlag, New York, 231 p.

Koenigsaecker, G., 2009. Leading the Lean Enterprise Transformation. CRC Press, Boca Raton, Florida, 121 p.

Kostoff, R.N. and Schaller, R.R., 2001. Science and Technology Roadmaps. IEEE Transactions on Engineering Management, vol. 48, no. 2, pp. 132-143.

Kurapati, N., Manyam, V.S.C. and Petersen, K., 2012. Agile software development practice adoption survey. Proceedings of the Agile Processes in Software Engineering and Extreme Programming Conference (XP), pp. 16-30.

Kuusela, R. and Koivuluoma, M., 2011. Lean Transformation Framework for Software Intensive Companies: Responding to Challenges Created by the Cloud. Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 378-382.

Lee, S. and Park, Y., 2005. Customization of technology roadmaps according to roadmapping purposes: Overall process and detailed modules. Technological Forecasting and Social Change, vol. 72, no. 5, pp. 567-583.

Leffingwell, D., 2007. Scaling software agility: best practices for large enterprises. Addison-Wesley, Boston, Massachusetts, 349 p.

Leffingwell, D., 2011. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Pearson Education, Boston, Massachusetts, 518 p.

Lehtola, L., Kauppinen, M. and Kujala, S., 2005. Linking the Business View to Requirements Engineering: Long-Term Product Planning by Roadmapping. Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE), pp. 439-446.

Lehtola, L., Kauppinen, M. and Vähäniitty, J., 2007. Strengthening the link between business decisions and RE: Long-term product planning in software product companies. Proceedings of the 15th IEEE International Requirements Engineering Conference (RE), pp. 153-162.

Lehtola, L., Kauppinen, M., Vähäniitty, J. and Komssi, M., 2009. Linking business and requirements engineering: is solution planning a missing activity in software product companies? Requirements engineering, vol. 14, no. 2, pp. 113-128.

Li, M. and Kameoka, A., 2003. Creating Added Value from Roadmapping Process: A Knowledge-Creating Perspective. Proceedings of the International Engineering Management Conference (IEMC), pp. 387-392.

Liker, J., 2004. The Toyota Way. McGraw Hill, New York, 330 p.

Lohan, G., Conboy, K. and Lang, M., 2010. Beyond Budgeting and agile software development: A conceptual framework for the performance management of agile software development teams. Proceedings of the International Conference on Information Systems (ICIS), pp. 1-13.

Lohan, G., 2013. A Brief History of Budgeting: Reflections on Beyond Budgeting, Its Link to Performance Management and Its Appropriateness for Software Development. Proceedings of the 4th International Conference on Lean Enterprise Software and Systems, pp. 81-105.

McCarthy, R.C., 2003. Linking Technological Change to Business Needs. Research Technology Management, vol. 46, no. 2, pp. 47-52.

Middleton, P. and Sutton, J., 2005. Lean Software Strategies. Productivity Press, New York, 432 p.

Middleton, P., Flaxel, A. and Cookson, A., 2005. Lean software management case study: Timberline inc. Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP), pp. 1-9.

Mintzberg, H., 1994. The fall and rise of strategic planning. Harvard business review, vol. 72, no. January-February, pp. 107-114.

Myers, K.L., 1999. CPEF: A continuous planning and execution framework. AI Magazine, vol. 20, no. 4, pp. 63-69.

Nardi, P.M., 2003. Doing survey research: a guide to quantitative methods. Pearson Education, Boston, 228 p.

Nordqvist, M. and Melin, L., 2010. The promise of the strategy as practice perspective for family business strategy research. Journal of Family Business Strategy, vol. 1, no. 1, pp. 15-25.

Nuseibeh, B. and Easterbrook, S., 2000. Requirements Engineering: A Roadmap. Proceedings of the Conference on the Future of Software Engineering, pp. 35-46.

Olsson, H.H., Alahyari, H. and Bosch, J., 2012. Climbing the "Stairway to Heaven"--A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 392-399.

Olsson, H.H., Bosch, J. and Alahyari, H., 2013. Towards R&D as innovation experiment systems: A framework for moving beyond agile software development. Proceedings of the IASTED International Conference on Software Engineering (SE), pp. 798-805.

Oppenheim, A.N., 1992. Questionnaire Design, Interviewing and Attitude Measurement. Printer Publishers, London, 303 p.

Overby, E., Bharadwaj, A. and Sambamurthy, V., 2005. A Framework for Enterprise Agility and the Enabling Role of Digital Options. Proceedings of the International Working Conference on Business Agility and Information Technology Diffusion Conference (IFIP TC8 WG 8.6), pp. 295-312.

Papatheocharous, E. and Andreou, A.S., 2014. Empirical evidence and state of practice of software agile teams. Journal of Software: Evolution and Process, vol. 26, no. 9, pp. 855-866.

Patton, M.Q., 2002. Qualitative Research & Evaluation Methods. Sage Publications, Thousand Oaks, California, 598 p.

Petersen, K., 2010. Is lean agile and agile lean? A comparison between two software development paradigms, in: Dogru, A.H. and Bicer, V. (Eds.), Modern software engineering concepts and practices: advanced approaches. Information Science Reference, New York, pp. 19-46.

Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M., 2008. Systematic mapping studies in software engineering. Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), pp. 68-77.

Phaal, R., Farrukh, C., Mills, J. and Probert, D., 2003. Customizing the Technology Roadmapping Approach. Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET), pp. 361-369.

Phaal, R., Farrukh, C. and Probert, D., 2000. Fast-Start Technology Roadmapping. Proceedings of the 9th International Conference on Management of Technology (IAMOT), pp. 1-12.

Phaal, R., Farrukh, C. and Probert, D., 2004a. Customizing Roadmapping. Research Technology Management, vol. 47, no. 2, pp. 26-37.

Phaal, R., Farrukh, C. and Probert, D., 2004b. Technology Roadmapping - A Planning Framework for Evolution and Revolution. Technological Forecasting and Social Change, vol. 71, no. 1-2, pp. 5-26.

Phaal, R., Farrukh, C. and Probert, D., 2005. Developing a Technology Roadmapping System. Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET), pp. 99-111.

Phaal, R., Farrukh, C. and Probert, D., 2009. Visualising strategy: a classification of graphical roadmap forms. International Journal of Technology Management, vol. 47, no. 4, pp. 286-305.

Phaal, R. and Muller, G., 2009. An architectural framework for roadmapping: Towards visual strategy. Technological Forecasting and Social Change, vol. 76, no. 1, pp. 39-49.

Phaal, R., Simonse, L. and Den Ouden, E., 2008. Next generation roadmapping for innovation planning. International Journal of Technology Intelligence and Planning, vol. 4, no. 2, pp. 135-152.

Pichler, R., 2010. Agile Product Management with Scrum: Creating Products that Customers Love. Addison-Wesley, Upper Saddle River, New Jersey, 133 p.

Poppendieck, M. and Poppendieck, T., 2003. Lean Software Development: An Agile Toolkit. Addison-Wesley, Boston, 203 p.

Poppendieck, M. and Poppendieck, T., 2007. Implementing Lean software development: From Concept to Cash. Addison-Wesley Professional, Upper Saddle River, New Jersey, 276 p.

Poppendieck, M. and Poppendieck, T., 2009. Leading Lean Software Development: Results Are Not the Point. Addison-Wesley Professional, Upper Saddle River, New Jersey, 278 p.

Radnor, Z. and Walley, P., 2008. Learning to walk before we try to run: adapting lean for the public sector. Public Money and Management, vol. 28, no. 1, pp. 13-20.

Rautiainen, K., Vuornos, L. and Lassenius, C., 2003. An Experience in Combining Flexibility and Control in a Small Company's Software Product Development Process. Proceedings of the International Symposium on Empirical Software Engineering (ISESE), pp. 28-37.

Rickards, R.C. and Ritsert, R., 2012. Rediscovering Rolling Planning: Controller's Roadmap for Implementing Rolling Instruments in SMEs. Proceedings of Economics and Finance, the 2nd Annual International Conference on Accounting and Finance (AF) and Qualitative and Quantitative Economics Research (QQE), pp. 135-144.

Ruhe, G., 2010. Product Release Planning: Methods, Tools and Applications. CRC Press, New York, 339 p.

Runeson, P. and Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, vol. 14, no. 2, pp. 131-164.

Shalloway, A., Beaver, G. and Trott, J.R., 2009. Lean-agile software development: achieving enterprise agility. Addison-Wesley Professional, Upper Saddle River, New Jersey, 262 p.

Staron, M., Meding, W. and Palm, K., 2012. Release readiness indicator for mature agile and lean software development projects. Proceedings of the Agile Processes in Software Engineering and Extreme Programming Conference (XP), pp. 93-107.

Strauss, J.D., Radnor, M. and Peterson, J.W., 1998. Plotting and navigating a non-linear roadmap: knowledge-based roadmapping for emerging and dynamic environments. Proceedings of the East Asian Conference on Knowledge Creation Management, pp. 1-26.

Suomalainen, T., Kuusela, R., Teppola, S. and Huomo, T., 2015. Challenges of ICT Companies in Lean Transformation. ACSIJ Advances in Computer Science: an International Journal, [Online], vol. 4, no. 2, No.14, pp. 49-56. Available from: http://www.acsij.org/documents/v4i2/ACSIJ-2015-4-2-671.pdf. [29.06.2016].

Tabrizi, B. and Walleigh, R., 1997. Defining next-generation products: an inside look. Harvard Business Review, vol. 75, no. 6, pp. 116-124.

Te Brömmelstroet, M., 2013. Performance of Planning Support Systems: What is it, and how do we report on it? Computers, Environment and Urban Systems, vol. 41, pp. 299-308.

Vähäniitty, J., Lassenius, C. and Rautiainen, K., 2002. An Approach to Product Roadmapping in Small Software Product Businesses. Proceedings of the 7th European Conference on Software Quality (ESCQ) - Quality Connection, pp. 12-13.

Van de Weerd, I., Bekkers, W. and Brinkkemper, S., 2010. Developing a Maturity Matrix for Software Product Management. Proceedings of the 1st International Conference on Software Business (ICSOB), pp. 76-89.

Van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J. and Bijlsma, L., 2006. Towards a reference framework for software product management. Proceedings of the 14th IEEE International Requirements Engineering Conference (RE), pp. 319–322.

Van Oosterhout, M., Waarts, E. and Van Hillegersberg, J., 2005. Assessing Business Agility: A Multi-Industry Study in the Netherlands. Proceedings of the International Working Conference on Business Agility and Information Technology Diffusion (IFIP TC8 WG 8.6), pp. 275-294.

Vidgen, R. and Wang, X., 2009. Coevolving systems and the organization of agile software development. Information Systems Research, vol. 20, no. 3, pp. 355-376.

VTT Technical Research Centre of Finland Ltd, VTT homepage. Available: http://www.vtt.fi/?lang=en [Referenced 27.06.2016].

Walsham, G., 1995. Interpretive case studies in IS research: nature and method. European Journal of information systems, vol. 4, no. 2, pp. 74-81.

Ward, A.C., 2007. Lean product and process development. Lean Enterprises Institute, Cambridge, 208 p.

Wareham, T.L. and Majka, A.J., 2003. Best practice financing. Kaufman Hall White Paper, Kaufman, Hall & Associates, Northfield, Illinois, pp. 1-35.

Wells, R., Phaal, R., Farrukh, C. and Probert, D., 2004. Technology Roadmapping for a Service Organization. Research Technology Management, vol. 47, no. 2, pp. 46-51.

Westkamper, E. and Von Briel, R., 2001. Continuous improvement and participative factory planning by computer systems. CIRP Annals-Manufacturing Technology, vol. 50, no. 1, pp. 347-352.

Wiegers, K.E., 2003. Software Requirements. Practical Techniques for Gathering and Managing Requirements throughout the Product Development Cycle. 2nd ed. Microsoft Press, Redmond, Washington, 546 p.

Willyard, C.M. and McCless, C.W., 1987. Motorola's technology roadmap process. Research Management, vol. 30, no. 5, pp. 13-19.

Womack, J.P. and Jones, D.T., 2003. Lean Thinking: Banish Waste and Create wealth in Your Corporation, Revised and Updated ed. Free Press, New York, 400 p.

Yin, R.K., 1994. Applied social research methods series Vol 5; Case Study Research: Design and Methods. 2nd ed. Sage Publications, Thousand Oaks, California, 171 p.

Yin, R.K., 2003. Case study research: design and methods. Sage Publications, Thousand Oaks; California, 181 p.

Appendix 1: Framework for the initial inquiry

The initial inquiry consists of both a questionnaire study and interviews. The framework for both research data collection methods are presented as follows.

Framework for the questionnaire study used in the initial inquiry

1. General Information

(By providing us your contact information you will receive a report from on questionnaire results)

Your name:	
Company:	
Department:	
Address:	
Title:	
Primary Role:	
Phone number:	
E-mail:	

2. Company Profile

What is the size of your company, i.e. total number of employees?

☐ <10 ☐ 10-49 ☐ 50-250

☐ >250 ☐ Other, what?

What is the life time of your company's products? In use by customers:

☐ 0-1 year ☐ 1-3 year ☐ 3-6 year

☐ 6-10 year ☐ >10 year ☐ Other, what?

3. Product Roadmapping Process

Who should participate in product roadmapping?

☐ Product management ☐ Finance

☐ Engineering ☐ Marketing

☐ Manufacturing ☐ Services

☐ Development ☐ Customer and partner representatives

☐ Other, what?

How many persons participate in a roadmapping process in your company?

- ☐ 1-5
- ☐ 6-10
- ☐ 11-20
- ☐ 21-30
- ☐ Other, what?

What is the most important phase in the roadmapping process? (Please rank most to least significant, 1=most significant)

- ☐ Capturing features into roadmaps
- ☐ Analysing features
- ☐ Prioritising features
- ☐ Roadmap validation and agreement
- ☐ Change management of the roadmap
- ☐ Other, what?

What is the most difficult phase in the roadmapping process? (Please select one)

- ☐ Capturing features into roadmaps
- ☐ Analysing features
- ☐ Prioritising features
- ☐ Roadmap validation and agreement
- ☐ Change management of the roadmap
- ☐ Other, what?

How are product requirements captured in roadmaps?

- ☐ With prototyping
- ☐ With interviews
- ☐ Gathering ideas over time (e.g. to a tool)
- ☐ In some kind of workshops, what kind?
- ☐ Other, what?

How are requirements prioritised in product roadmapping?

- ☐ With formal methods
- ☐ With informal methods(Please describe the method / practice)

If formal methods are used for requirements prioritisation, what are the formal methods?

- ☐ Analytical Hierarchy Process (AHP)
- ☐ Quality Function Deployment (QFD)
- ☐ EVOLVE
- ☐ Distributed Prioritisation
- ☐ Other, what?

If your company has experience of product roadmapping in collaboration, please select the used collaboration mode.

- ☐ Joint research and development partnership
- ☐ Customer–supplier relationship
- ☐ Technology exchange agreement or licensing
- ☐ Other, what?

In your opinion, did the collaboration affect the roadmapping process?

- ☐ No
- ☐ Yes (Please describe more details)

Would you be willing to participate in an interview relating to product roadmapping?

- ☐ No
- ☐ Yes, select the most suitable practice
- ☐ Phone interview (max. duration ½ hour)
- ☐ Face-to-face interview (max. duration 1 hour)

Framework for the Interviews used in **the initial inquiry**

Product Roadmapping Process

- How does the roadmapping process begin?
- What roles does the roadmapping process include?
- Who participates into the roadmapping process and to which activities in it?
- What do they do during the roadmapping process (what are their responsibilities and what viewpoints are their concern)?
- How does the roadmapping process proceed? / What are the main phases of the product roadmapping process?

Collaboration viewpoints

- What are the most important activities in roadmapping in a collaboration situation?
- How is communication (e.g. about requirements, and change management) arranged between collaboration partners?

- How are roadmaps created together with collaboration partners (e.g. participants, responsibilities, tasks, decision-making, gaining mutual understanding)?
- How does the collaboration change the product roadmapping process / the content of the product roadmap?

Capturing features
- How are features captured?
- How does the collaboration mode affect capturing features?

Analysing features
- How are features analysed?
- What methods / practices are used when analysing features?
- How does the collaboration mode affect analysing features?

Prioritising features
- How are prioritisation methods selected (between collaboration partners)?
- How does the selected prioritisation method support collaboration (especially: AHP, QFD, Distributed Prioritisation, EVOLVE)?
- How are requirements (functional and especially non-functional requirements) prioritised when there are several stakeholders?
- How does the collaboration mode affect prioritising product features?
- Who makes the final decision about the priorities?

Roadmap validation and agreement
- How are roadmaps validated?
- How is the roadmap agreement made?
- How does the collaboration affect the roadmap validation and agreement?

Change Management of the product roadmaps
- How are changes to product roadmaps managed in collaboration networks?
- Who should participate in the impact analysis?
- How is the impact analysis carried out in collaboration environment?
- Who makes the final decision concerning the change in collaboration?

What are the benefits of product roadmapping?
What are the problems of product roadmapping?

Appendix 2: Themes and questions of the research interviews

The original themes for the research interviews used both in semi-structured interviews and in narrative interviews are listed as follows.

Semi-structured interview themes

Background of the interviewee

- Name:
- Role:
  - Role relating to continuous planning:
  - Responsibilities relating to continuous planning:
- Organisation unit:
  - With whom do you collaborate with/how/what?
    - *Organisation*
- Could you draw a picture of the case company organisation?
- What are the levels of planning?
- Who is involved?
- Who makes the decisions at these levels?
  - How are you involved in decision making?
- How is customer feedback collected and included in planning?
- What are the used practices/methods/tools for working and communication?

  For example:
  - Are lean and agile practices/methods/tools used?
  - Are Continuous Deployment practices/methods/tools used? What and How?
  - What is their relation to planning?
  - How does information flow proceed and how is information distributed?

1. Definition of continuous planning (CP)

- What does continuous planning mean (in our company or to you)?
- What does transparency mean to you – how it is related to CP?

2. Background for CP: Why CP?

- Why did you adopt / end up with continuous planning?
- What were the problems / drawbacks in the previous ways of planning?

- Could you draw a timeline for adopting CP? For example:
  - When have you started continuous planning?
  - When are you expecting to see the results of continuous planning?

- What are the issues that you have needed to do / change in order to achieve continuous planning?

Continuous Planning Process

3. Levels of planning and time frame
   - What are the levels of planning in your company?
     - To which level(s) does continuous planning relate to?
   - What is the time frame for each level of continuous planning?
   - How often is continuous planning done / what is the cycle of planning?
   - What triggers continuous planning?
     - When are the plans updated? Why?
4. What is the continuous planning process?
   - How do you conduct continuous planning (current status) AT EACH LEVEL?
     → Could you show or draw a picture about it?
   - What are the main phases of continuous planning?
     → Could you specify these phases in more detail?
   - How / in which situation is continuous planning launched /started?
   - How does the continuous planning process continue?
   - How often is CP done?
   - How is the status of the plans followed (progress vs. plans & updates)?
   - How does continuous planning relate to other levels of planning?
   - What kinds of methods for continuous planning are used?
   - What kinds of tools for CP are used?
   - Where are the plans stored?
   - Who sees these plans?
   - How can feedback to plans be given?

Continuous Planning Participants

5. Who participates in continuous planning?
   - Who participates in continuous planning (at each level)?
     → E.g. who are the stakeholders, internal and external?
   - How many participants are involved?
   - What are the main roles?
   - What are the main tasks?
   - Who makes decisions relating to the plans?
   - How are the other levels of plans visible to the participants?

Continuous Planning Lessons Learned:

6. Expectations, Benefits, Challenges, Enablers and Obstacles and Impacts
- Expectations: What do you expect to achieve with continuous planning?
- Benefits: What are the benefits of continuous planning?
  ➔ What works especially well in your organisation?
- Risks / Challenges: What are the risks/challenges relating to continuous planning?
  ➔ What causes these risks / challenges?
  ➔ How have you tried to solve these problems?
- Obstacles: What are the obstacles relating to continuous planning?
  ➔ What things are affecting e.g. to CP adoption?
- Impacts: What are the impacts of continuous planning?

- What are the improvement suggestions for adopting CP?
  → How can companies improve their continuous planning practices?

Competencies and Transparency:

7. Competence Development, Learning, and Transparency
- What has been required so that the organisation can use continuous planning?
  - Has it required some new skills/tools? What kinds of skills/tools?
  - Have the organisation / the organisation structures been changed, and how?
  - Have the ways of working been changes, and how?
  - How have competencies bent to / grown through continuous planning?
- Transparency: How important is transparency in your organisation?
  - How development transparency is implemented in your organisation?
  - How learning is improved?

Narrative interview themes

Could you describe your current position and background in the case company?

Issues that could come up in the narrative:

- Role/ title, work description, responsibility
- When started working, previous roles/tasks, etc.
- Role especially from the planning perspective

What kind of an organisation is the case company? How does the case company work as an organisation?

Issues that could come up in the narrative:

- Draw a picture of the case company organisation
- With whom do you collaborate with/how/what?
- How does information flow proceed and how is information distributed?

- Used practices/methods/tools for working and communication (e.g. are lean and agile practices/methods/tools used, their relation to planning?)

How are operations planned in the case company? How is planning done in practice?

Issues that could come up in the narrative:

- How does planning start/begin (e.g. what is the need/trigger at the background)?
- How is planning conducted in practice e.g. what are the tools/methods for planning (are e.g. continuous planning practices used)?
- What are the (main) phases /stages / levels of planning (e.g. the whole organisation wide strategy, business areas, product portfolio, products, features, etc.)?
- How do the different plans and levels of plans communicate with each other?
- What is the time frame(s) of planning /plans (e.g. current situation, short-term and long-term plans)?
- What triggers continuous planning OR when are the plans updated? Why?
- Who participates in the planning; main roles, tasks, decision making?
- With whom are the plans created (naming the key contact persons)? Do other participants outside the organisation also participate in the planning (e.g. customers)?
- How is information used to support planning, e.g. is customer data used somehow, if yes how and where is the customer data collected /received from?
- How are the plans taken into practice? How is the information from the plans / planning sessions distributed?
- Have the planning methods/ practices been changed recently, if yes how?
- Current practices:
    - Strengths (benefits)
    - Weaknesses (risks and problems)
    - Goals (if there is a need for the current practices to be changed, e.g. improvement suggestions?)

Appendix 3: List of interviews conducted for the study

Interviews conducted for the initial inquiry:

Date	Duration	Company ID	Role of the interviewee	Interview method
07.09.2006	35 min	A	Manager	Semi-structured phone interview
07.09.2006	68 min	A	Group manager	Semi-structured face-to-face interview
07.09.2006	50 min	B	Chief technology officer	Semi-structured phone interview
07.09.2006	31 min	C	Product planner	Semi-structured phone interview
08.09.2006	30 min	D	Program director	Semi-structured phone interview
08.09.2006	49 min	E	Group manager	Semi-structured phone interview
08.09.2006	65 min	F	Chief engineer	Semi-structured face-to-face interview
13.09.2006	15 min	G	Senior researcher	Semi-structured phone interview
13.09.2006	61 min	H	Manager	Semi-structured phone interview

Interviews conducted for the case A:

Date	Duration	Role of the interviewee	Interview method
27.11.2012	77 min	Head of quality and environment	Semi-structured face-to-face interview
21.11.2014	86 min	Scrum master	Semi-structured face-to-face interview
21.11.2014	90 min	Team leader / project manager	Semi-structured face-to-face interview
26.11.2014	93 min	Sales director	Semi-structured face-to-face interview

27.11.2014	92 min	Product manager	Semi-structured face-to-face interview
4.2.2015	67 min	President of business segment	Semi-structured face-to-face interview
5.11.2014	122 min	2 Quality managers	Semi-structured face-to-face interview
23.3.2015	78 min	Business developer	Semi-structured face-to-face interview
28.4.2015	78	Vice president of business area	Semi-structured face-to-face interview
17.6.2015	74 min	Sales director	Semi-structured face-to-face interview

Interviews conducted for the case B:

Date	Duration	Role of the interviewee	Interview method
08.06.2011	108 min	Project manager	Semi-structured face-to-face interview
16.10.2014	79 min	Senior product marketing manager	Narrative face-to-face interview
24.10.2014	71 min	Senior product marketing manager	Narrative face-to-face interview
24.10.2014	81 min	Chief strategy officer	Narrative face-to-face interview
27.10.2014	49 min	Product marketing management	Narrative face-to-face interview
27.10.2014	68 min	Senior product manager	Narrative face-to-face interview
6.11.2014	68 min	Product marketing management	Narrative face-to-face interview
6.11.2014	57 min	Director of product management	Narrative face-to-face interview
6.11.2014	69 min	Director of product management	Narrative face-to-face interview
7.11.2014	64 min	Director of product management	Narrative face-to-face interview
10.11.2014	50 min	Executive vice president	Narrative face-to-face interview
10.11.2014	57 min	Senior product manager	Narrative face-to-face interview
11.11.2014	41 min	Vice president of R&D	Narrative face-to-face interview

Interviews conducted for the case C:

Date	Duration	Role of the interviewee	Interview method
21.01.2015	55 min	Line manager	Semi-structured face-to-face interview
21.01.2015	62 min	Domain manager	Semi-structured face-to-face interview
21.01.2015	79 min	Product owner	Semi-structured face-to-face interview

Appendix 4: List of interviewee profiles

Summary of interviewee profiles in the initial inquiry

Interviewee	Company Nationality	Company ID	Company size (employees)	Role of the Interviewee
1	Finnish	A	> 250	Manager
2	Finnish	A	> 250	Group Manager
3	Finnish	B	50–250	Chief Technology Officer (CTO)
4	Finnish	C	> 250	Product planner
5	Finnish	D	> 10	Program Director
6	Finnish	E	> 250	Group Manager
7	Finnish	F	> 250	Chief Engineer
8	Swedish	G	> 250	Senior Researcher
9	Finnish	H	50–250	Manager

Summary of interviewee profiles in the multiple-case study

Interviewee	Company Nationality	Company ID	Company size (employees)	Role of the Interviewee
1	Finnish	Case A	~1800	Head of quality and environment
2	Finnish	Case A	~1800	Scrum master
3	Finnish	Case A	~1800	Team leader / project manager
4	Finnish	Case A	~1800	Sales director
5	Finnish	Case A	~1800	Product manager
6	Finnish	Case A	~1800	President of a business segment
7	Finnish	Case A	~1800	2 Quality managers
8	Finnish	Case A	~1800	Business developer
9	Finnish	Case A	~1800	Vice president of a business area

10	Finnish	Case A	~1800	Sales director
11	Finnish	Case B	~1000	Project manager
12	Finnish	Case B	~1000	Senior product marketing manager
13	Finnish	Case B	~1000	Senior product marketing manager
14	Finnish	Case B	~1000	Chief strategy officer (CSO)
15	Finnish	Case B	~1000	Product marketing management
16	Finnish	Case B	~1000	Senior product manager
17	Finnish	Case B	~1000	Product marketing management
18	Finnish	Case B	~1000	Director of product management
19	Finnish	Case B	~1000	Director of product management
20	Finnish	Case B	~1000	Director of product management
21	Finnish	Case B	~1000	Executive vice president
22	Finnish	Case B	~1000	Senior product manager
23	Finnish	Case B	~1000	Vice president of R&D
24	Swedish	Case C	~110 000	Line manager
25	Swedish	Case C	~110 000	Domain manager
26	Swedish	Case C	~110 000	Product owner

PAPER I

Challenges for Product Roadmapping in Inter-company Collaboration

Proceedings of the Third International Conference on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD), pp. 66–80. ETH Zurich, Switzerland on July 2–3, 2009.

PAPER II

Software Product Roadmapping in a Volatile Business Environment

The Journal of Systems and Software,
Vol. 84, Issue 6, pp. 958–975.

The Journal of Systems and Software 84 (2011) 958–975

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Software product roadmapping in a volatile business environment

Tanja Suomalainen [a,*], Outi Salo [b], Pekka Abrahamsson [c], Jouni Similä [d]

[a] VTT Technical Research Centre of Finland, Department of Software Technologies, P.O. Box 1100, FI-90571 Oulu, Finland
[b] Nokia Corporation, P.O. Box 300, FI-90401 Oulu, Finland
[c] Free University of Bozen-Bolzano, Italy
[d] Department of Information Processing Science, University of Oulu, Finland

ARTICLE INFO

Article history:
Received 27 August 2010
Received in revised form
17 December 2010
Accepted 13 January 2011
Available online 20 January 2011

Keywords:
Product roadmapping
Roadmapping process
Product development
Software product management

ABSTRACT

Product roadmapping enhances the product development process by enabling early information and long-term decision making about the products in order to deliver the right products to the right markets at the right time. However, relatively little scientific knowledge is available on the application and usefulness of product roadmapping in software product development context. This study develops a framework for software product roadmapping, which is then used to study the critical aspects of the product roadmapping process. The collection of empirical evidence includes both quantitative and qualitative data which sheds further insight into the complexities involved in product roadmapping. Results revealed that organizations view the product roadmap mainly as a tool for strategic decision making as it aims at showing the future directions of the company's products. However, only a few companies appear to have an explicit approach for handling the mechanisms for creating and maintaining such a roadmap. Finally, it is suggested that the strategic importance of product roadmapping is likely to increase in the future and, as a conclusion, a new type of agility is required in order to survive in the turbulent and competitive software business environment.

1. Introduction

Nowadays, the cost, complexity and rate of technology change are increasing while competition and sources of technology are becoming more global (Phaal et al., 2000a,b). Companies are expected to be more responsive to technological change and to manage their technology assets more strategically because corporate strategies often lack influence over or contact with those who directly manage products (Kappel, 2001). The business environment is characterized by ever-more-demanding customers, increasingly shorter product life cycles and rapidly developing technologies (Groenveld, 2007). The demand for an overall view of the product and its future releases has become important (Lehtola et al., 2005). Therefore, developing reliable and high quality software products on time and within budget requires a well-coordinated and executed software process (Jiang and Coyner, 2000).

Product roadmapping is an approach to manage a high-level view and to link aspects of business to requirements engineering (RE) (DeGregorio, 2000). It aims at enabling developments in technology to be mapped and linked to product evolution and market opportunities (Phaal et al., 2003a). Thus, roadmapping aims at enhancing two main purposes in the context of software engineering: how to serve important markets with the right products at the right time and how to improve the cross-functional processes required for the creation of new products (Groenveld, 2007). Several benefits have been claimed (McCarthy, 2003) to result from the application of roadmapping in organizations. For one, it has been claimed as a simple process for the presentation and cross-functional understanding of complicated issues. It has also been proposed (McCarthy, 2003), for example, to enable a faster and superior assessment of emerging technologies and the contents of products from the learning obtained during the roadmapping process. Also, some difficulties relating to roadmapping have been identified. Yoon et al. (2008) have expressed that many companies, especially medium and small companies, have difficulties in implementing and sustaining roadmapping due to factors such as maintaining the time, cost and effort estimations. A lack of quality input data on markets, competitors and technology, gathered from workshops, complicates the adoption of the method. Furthermore, regular updating of roadmaps can be long and tedious, and may require further workshops and repletion of the roadmapping process (Yoon et al., 2008).

However, currently relatively little scientific knowledge is available on the usefulness and application of product roadmapping in software product development organizations. Research findings

* Corresponding author. Tel.: +358 405122610; fax: +358 207222320.
E-mail addresses: Tanja.Suomalainen@vtt.fi (T. Suomalainen), Outi.Salo@nokia.com (O. Salo), Pekka.Abrahamsson@unibz.it (P. Abrahamsson), Jouni.Simila@oulu.fi (J. Similä).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.01.031

presented in this paper aim at defining a product roadmapping process and by so doing, increase product roadmapping knowledge for both science and industry. From industry viewpoint, this study aims at increasing the product roadmapping knowledge in order to assist companies in developing their own product roadmapping activities. From scientific perspective, the study aims at increasing the body of knowledge in the area of product roadmapping while also identifying future research and development work opportunities on the most critical aspects of the product roadmapping process. The research is structured and presented by using a research framework for software product roadmapping, which is created based on the current literature. The research framework provides the lenses that guide the empirical analysis. These lenses are defined as follows: defining the main stakeholders and their roles during the process, organizing the product roadmapping process, establishing the main benefits and challenges faced during the process, and identifying the most critical phases of the process. The research provides empirical evidence from altogether nine interviews and a questionnaire obtained from 34 different companies.

This article is structured as follows: in section two, product roadmapping is defined based on the current literature including an overview of roadmapping, stakeholders of the product roadmapping process and product roadmapping phases. In section two, a software product roadmapping framework is also presented based on the existing literature including the central elements of roadmapping, and the perceived impacts of software product roadmapping and its main phases. Thereafter, in section three, the research design of this study is presented, which includes the research methods and goals as well as the research context and data collection methods. In section four, the research evidence from the empirical study is presented and aligned with the presented research framework. In section five, the research results are discussed and, finally, in section six, conclusions are drawn.

2. Related work

In this section, we will outline the current knowledge on product roadmapping based on the existing literature and research. Additionally, in this section, a research framework for software product roadmapping is proposed based on the existing literature. Firstly, the terminology of roadmapping is introduced (Section 2.1). Then, stakeholders (Section 2.2) of product roadmapping, and the proposed main phases (Section 2.3) of product roadmapping are specified. Finally, a research framework for software product roadmapping (Section 2.4) is presented.

2.1. Overview

A roadmap is a layout of existing routes or paths, which is used to decide among alternative directions towards a desired destination (Kostoff and Schaller, 2001). It is said by DeGregorio (2000) that a roadmap is a visualization of a forecast, which can address a number of key areas, such as technology, capability, platform, system, environment, threat, and business opportunity. According to Kappel (2001), roadmaps are also forecasts of what is possible or likely to happen, and plans that express a course of action. Furthermore, roadmaps are intended to be living documents, to be reviewed and updated over time in order to remain useful (Albright, 2003). Roadmapping describes the process of creating and revising roadmaps (Kostoff and Schaller, 2001). It is a strategic planning and forecasting process with long-lasting future activities (Kappel, 2001). Roadmapping is also claimed (Li and Kameoka, 2003) to be a decision-making and design process. On the other hand, Strauss et al. (1998) describe roadmapping as a management activity that links customer/market needs and opportunities, product quality and competitive positioning as well as corporate capabilities such as business and technology value chain attributes.

Roadmapping can be seen similar to strategic planning. However, strategic planning does not mean necessarily strategic thinking, and thus it is claimed by Mintzberg (1994) that the most successful strategies are visions, not plans. When the difference between planning and strategic thinking is understood, the company can get back to the strategy making process: capturing what the manager learns from all sources. Furthermore, according to Mintzberg (1994) strategic planning is about breaking down a goal or a set of intentions into steps, formalizing those steps to be implemented almost automatically, and then articulating the anticipated consequences or results of each step. On the contrary, strategic thinking is about synthesis and involving intuition and creativity. The strategic thinking results in an integrated perspective of the company, but not too precisely articulated vision of direction (Mintzberg, 1994). For example, Beeton et al. (2008) have classified roadmapping strategies into two categories. Either the roadmapping may be exploratory, i.e. surveying future possibilities, or goal-oriented, i.e. defining strategies to realise clearly defined future targets.

Roadmaps can be expressed in various forms, types or with different taxonomies (Kameoka et al., 2003). For example, according to Kappel (2001) roadmaps can be categorized into science or technology roadmaps, industry roadmaps, product-technology roadmaps, and product roadmaps. Kappel (2001) describes the interrelationships between these four types of roadmaps as follows. The purpose of the science or technology roadmaps is to set industry targets and understand the future by identifying trends and making accurate forecasts. The industry roadmap is a combination of technology forecasts and industrial matters. Thus, the purpose of the industry roadmap is to set industry expectations that express for example a technical thrust and a competitive environment. The product-technology roadmap is a combination of specific product plans with technology trends and marketplace. The purpose of the product-technology roadmaps is to align product and technology generations together. On the other hand, the product roadmaps schedule product introductions, such as a direction and schedule for product evolution, which are then communicated with customer and internal audience (Kappel, 2001).

The taxonomy according to Kappel (2001) is presented in a chart, in which the horizontal axis describes the roadmapping purpose on the industry or company level, and the vertical axis describes the content emphasis of the roadmap either on specific trends or on positioning within an industry. In addition, Phaal et al. (2005) claim that roadmaps commonly take a form of a multi-layered time based chart that includes different layers of knowledge relating to purposes, deliveries, and resources. On the contrary, Vähäniitty et al. (2009) state that the product roadmap consists of five layers with the four topmost depicting activities and the bottom layer illustrating the estimate of human resource requirements. The activities are such as performing services, preparing releases, developing product components and platforms. According to Albright (2003) and Groenveld (2007) the multi-layered roadmap can be constructed from a market-pull or from a technology-push perspective. From the market-pull point of view, a roadmap should begin with defining the most important requirements of the marketplace and customers. This strategy includes defining product development in the process of time and defining the required technologies for these products. From the technology-push viewpoint, a roadmap should begin with defining the key or new technologies and their market needs. This strategy describes how technology is going to affect the functionality of the product (Albright and Kappel, 2003; Groenveld, 2007). Even though roadmaps may take various forms or taxonomies, they all should answer a common set of "why-what-how-when" questions that generally relate to markets,

products, and technologies (Phaal et al., 2005). However, as Phaal et al. (2004) emphasize, the form of the roadmap should be tailored to the specific needs of the company and its business context.

A product roadmap provides a forecast of product family evolution over time and views the whole platform or relationships between the products in a platform (Albright and Kappel, 2003). Typically, product roadmaps are developed, reviewed and improved iteratively. Often, this is achieved through close human interaction such as face-to-face meetings and workshops with relevant stakeholders (Phaal et al., 2005). Typically, product roadmaps are owned by the business owner of the product, who is also responsible for gathering all relevant stakeholders to obtain the needed information for the roadmaps. Product roadmaps typically cover a scope of two to three years, during which time they are to be frequently revised in order to ensure the currency of the documentation (Tabrizi and Walleigh, 1997; Lehtola et al., 2005). Documentation of features is an important part of product roadmapping, since accurate documentation ensures that the roadmaps can be read, analysed, redrawn, and validated (Nuseibeh and Easterbrook, 2000). The information in the product roadmaps incrementally describes how the product and its business environment changes yearly. The fields in the product roadmap define the high-level functionality of the product and target customer group. The high-level product functionality is a description of forthcoming releases with basic mandatory information. The mandatory information includes the release goal and high-level features for each release. In addition, release time, localization, platforms, and dropped topics, e.g. features that are not supported in the subsequent versions of the product, are recorded. Additionally, the fields include information about positioning, market arguments, and the geographical focus of the product for every year. This information is represented with a few bullet points for each issue (Lehtola et al., 2005).

Product roadmapping pertains to the software engineering (SE) process as it is a method for planning and defining software product requirements based on the market and needs of stakeholders. Software systems requirements engineering (RE) is defined as a process of discovering a system's purpose, by identifying stakeholders and their needs, and documenting these in a form that is adequate to analysis, communication, and subsequent implementation (Nuseibeh and Easterbrook, 2000). Product roadmapping is also closely linked to RE. For example, according to Jantunen and Smolander (2006) product roadmapping deals with the different releases of each product. In a product roadmap, the product is represented as product releases containing several product features. According to Wiegers (2003), a product feature is a set of logically related requirements that provide a capability to the user and enable the satisfaction of a business objective. Instead, a requirement is a statement of a customer need or objective. A requirement can also be a condition, which a product must meet to satisfy such a need or objective. In other words, a requirement is a property that a product must have to provide value to a stakeholder. However, product roadmapping differs from RE in that it is a process with long-lasting future activities, and in roadmapping, high-level features are presented within a timeline and scheduled for different releases. On the contrary, in RE, the features are analysed in more detail and defined as to what they mean from the perspective of the product development project. Also, Lehtola et al. (2005) have pointed out that targets for the release roadmapping and the product roadmapping are quite the same but the product roadmapping is more high-level. The main objective of the release roadmapping is to inform stakeholders about scheduled future releases. Instead, the objective of the product roadmapping is to help product managers create and maintain release roadmaps, manage situations where the same technical product is included in several products, and R&D to identify the needs for research projects (Lehtola et al., 2005). Furthermore, product roadmapping can be seen as a part of broader perspective to software product management. According to a reference framework for software product management (van de Weerd et al., 2006), the product management consists of portfolio management, product roadmapping, requirements management, and release planning. In that framework, product roadmapping includes theme identification, core asset identification, and roadmap construction. Instead, requirement management includes activities of gathering, identifying and organizing requirements, and release planning includes activities such as requirements prioritisation, selection, definition, validation, scope change management and launch preparation.

2.2. Product roadmapping stakeholders

It has been suggested (Tabrizi and Walleigh, 1997) that product roadmaps are created by senior management who are also responsible for updating the roadmaps. However, it has been argued (Li and Kameoka, 2003; Lehtola et al., 2005) that the roadmapping team more usually consists of various stakeholders from different functions of the organization or even from different organizations. The team shares information and perspectives to make decisions that are then presented in a roadmap (Lehtola et al., 2005). The study of Lehtola et al. (2005) supports the latter approach, though it does not propose who the participants are, and what their role in the roadmapping process is. Nevertheless, the research of Lehtola et al. (2005) indicates that the most important stakeholder groups to which the contents of the roadmaps should be communicated and with which they should be negotiated were product management, sales and channel partners, and customers. The product developers were not seen as an important stakeholder group in the roadmapping process, yet they were considered important in estimating the costs of future requirements.

According to McCarthy (2003), only the roadmapping team participates in the roadmapping process while support from management is needed regarding personnel and budget investments. The team should be formed at the beginning of the roadmapping process, including the research and development (R&D) and technology management personnel, members from business development, representatives from finance, and core staff members from the other functions. The first task of the team is to establish a common understanding of the process and the terminology to be used. After that, the team should begin to develop a detailed analysis of the process, and to decide factors and metrics required for the process evaluation. The roadmapping team is also responsible for analysing the required technologies as well as implementing and reviewing the roadmaps.

On the other hand, Groenveld (2007) proposes that the roadmapping process should be started with a small roadmapping team in which the marketing, product management, research, development, and engineering teams participate. Later, the team looks for a leader who should become the owner of the drafted roadmaps. The owner is responsible for the maintenance of the roadmaps and for the initiation of appropriate updating actions as well as providing additional information when needed. The roadmapping team guides the process and organizes workshops to ensure integral involvement of the organization and input by the organization. The outcome of the workshops is used to prepare draft roadmaps, or parts of them.

According Phaal et al. (2000a,b, 2003a), a multifunctional team is needed in the roadmapping process in order to provide multiple perspectives such as commercial, technical, research, development, manufacturing, marketing, and finance. In addition, Phaal et al. (2003a) believe that both the business owner and the process owner should participate in the roadmapping process. The owners should be involved in the planning phase and, thereafter, through-

Table 1
Summary of the roadmapping process stakeholders.

Stakeholders	Reference							
	Tabrizi and Walleigh (1997)	Lehtola et al. (2005)	McCarthy (2003)	Groenveld (2007)	Phaal et al. (2000a,b), Phaal et al. (2003a)	Albright (2002)	Rautiainen et al. (2003)	van de Weerd et al. (2010)
Senior management	X						X	
Product management		X		X		X	X	X
Sales and channel partners		X						X
Customers		X					X	X
Representatives of R&D			X	X	X	X	X	X
Technology management			X					
Representatives of business development			X					
Representatives of finance			X		X			
Representatives of marketing				X	X	X		X
Representatives of engineering				X				
Representatives of manufacturing					X	X		
Representatives of services						X	X	X
Representatives of support								X

out the roadmapping process. The business owner is proposed to be responsible for the business outcome of the process, while the process owner is responsible for the implementation of the roadmapping (Phaal et al., 2003a; Wells et al., 2004). The owners are also responsible for selecting the persons to the roadmapping team, solving issues regarding the application, and having knowledge about the roadmapping domain (Phaal et al., 2003a). In addition, a facilitator is proposed for managing and facilitating the roadmapping process (Phaal et al., 2003a; Wells et al., 2004). During the different phases of the roadmapping, a facilitator can support and guide the roadmapping team (Albright, 2002). Moreover, according to Albright and Kappel (2003), facilitators have active roles in appropriately scoping the roadmap, forming the team, setting up a work plan, and assessing individuals with their tasks in the larger effort. The facilitator should also challenge assumptions and force rigour into the roadmap (Albright and Kappel, 2003).

Albright (2002) also suggests that the roadmapping process is best performed as a cross-functional team led by an experienced facilitator. Through the whole process, the facilitator steers the team towards a realistic plan. The cross-functional team includes many functions that contribute to the success of a product line or business: central and regional marketing, product management, R&D, manufacturing, services, etc. The purpose of the roadmapping team is to lay out a possible future or multiple futures, set objectives, and define a plan to achieve the objectives, as well as make sure that the required capabilities and technologies are available at the right times (Albright, 2002). Furthermore, Jantunen and Smolander (2006) have identified three types of roles that need to be present in a roadmapping process: contributor, controller, and distributor. Contributor is the one that brings valuable information to a roadmapping context. Controller ensures that roadmapping is done systematically, and distributor absorbs information at a roadmapping context and disseminates the results of roadmapping to those who need to act upon it. Contributor has similar tasks as the previously mentioned member of a roadmapping team, and controller acts in a quite similar role as a facilitator.

van de Weerd et al. (2006, 2010) have made a difference between internal and external stakeholder groups. Their definition of the product management stakeholders is adopted from (Gorchels, 2000; Lehtola et al., 2005). The internal stakeholders are described as follows: product management, company board, research and innovation, services, development, support, and sales and marketing. The external stakeholders are defined as follows: market (i.e. potential customers, competitors, and analysts), partners (i.e. implementation, development, and distribution partners), and customers.

Clearly, in literature there are differing views on how widely and in what role relevant stakeholders should participate in the roadmapping process. Just as clearly, the different views are largely based on opinions or views and not on empirical research about how the roadmapping process is actually carried out in practice and what kinds of problems and challenges have been met in practice. Table 1 provides a summary of the stakeholders of the roadmapping process presented by different authors.

2.3. Product roadmapping process

The roadmapping process focuses on sharing perspectives, involving interaction between people, leading to communication, new understanding, insights, creativity, and learning (Phaal et al., 2005). However, due to the different markets and cultures of companies the roadmapping process varies between companies (Phaal et al., 2004; Groenveld, 2007). According to Phaal et al. (2004), finding a suitable roadmapping process depends on many factors such as the level of available resources (e.g., people and time), issues to be addressed (e.g., purpose and scope), information available (e.g., market and technology), as well as other existing processes and management methods (e.g., new product development, project management and market research).

Actually, only a few process descriptions focusing on product roadmapping seem to have been proposed in the literature. Albright and Kappel (2003) define the roadmapping process to include stages of initiation, maintenance, and restart, based on an organisation's specific needs. Phaal et al. (2003a) identify three roadmapping phases: planning, facilitated roadmapping workshop(s), and roll-out. A fairly similar three-phase product roadmapping process has been proposed also by Lehtola et al. (2005) including preparation, approval, and communication. On the contrary, van de Weerd et al. (2010) define product roadmapping process to consist of theme identification, core asset identification, and roadmap construction. Alternately, McCarthy (2003) adds two more phases to the roadmapping process that consists of team formation, focus, technology or workflow analysis, implementation, and review. Moreover, Vähäniitty et al. (2002) propose a four-step model especially for creating and updating product roadmaps. The following steps are suggested: (1) Define the strategic mission and vision of the company, and outline the product vision. (2) Scan the environment by identifying major trends. (3) Revise and distil the product vision as product roadmaps. (4) Estimate the product life cycle and evaluate the mix of development efforts planned. The steps in the model should be performed periodically to adjust the roadmap to new information

Table 2
Summary of roadmapping processes.

Name	Goal	Main phases	Reference
Roadmapping life cycle	Goal is to define and communicate product and technology strategy along with a longer, smarter view of the future	Initiation Maintenance Restarts	Albright and Kappel (2003)
T-Plan: fast-start technology roadmapping (TRM)	Goal is to bring together key stakeholders and experts to capture, share and structure knowledge about the issue being addressed, to identify strategic issues and to plan the way forward	Planning Facilitated workshop(s) Roll-out	Phaal et al. (2003a)
Product roadmapping (PRM)	Goal is to help (1) product managers create and maintain release roadmaps, (2) managing situations where the same technical product is included in several products, and (3) R&D to identify the needs for research projects	Preparation Approval Communication	Lehtola et al. (2005)
Release roadmapping (RRM)	Goal is to inform stakeholders about scheduled future releases to help R&D, for example, plan their skills development and act as a trigger for early feature development, or marketing, to plan their future activities	Data collection Feature prioritisation Release planning Release roadmap validation	Lehtola et al. (2005)
Roadmapping process	Goal is to improve internal processes which may need improvement to increase R&D productivity or to upgrade a step in the drug discovery process that has fallen behind “industry standards”	Team formation Focus Technology/workflow analysis Implementation Review	McCarthy (2003)
Four-step model for creating and updating product roadmaps	Goal is to define and concretise the company’s plans for technology and product development	Define strategic mission and vision, and outline product vision Scan the environment Revise and distil the product vision as product roadmaps Estimate product life cycle and evaluate the mix of development efforts planned	Vähäniitty et al. (2002)
Product roadmapping	Goal is to handle the development of the product roadmap, in which the future releases are planned based on themes and core assets	Theme identification Core asset identification Roadmap construction	van de Weerd et al. (2010)

and changing market situations. Smaller updates to the roadmaps are suggested to ensure up-to-date information (Vähäniitty et al., 2002).

However, unlike the others, Lehtola et al. (2005) separate the release roadmapping process as distinct from product roadmapping. It is proposed to consist of data collection, feature prioritisation, release planning, and release roadmap validation. Similarly, van de Weerd et al. (2010) separate the requirements management and release planning as distinct from product roadmapping. The requirements management consists of gathering, identifying, and organizing requirements. The release planning includes requirements prioritization, requirements selection, release definition, release validation, launch preparation, and scope change management. In Table 2, a summary of the existing roadmapping processes is presented including their main focus areas, i.e. goals and main phases.

2.3.1. Capturing features into roadmaps

According to van de Weerd et al. (2010), the product roadmapping starts with identifying themes and core assets for the future releases. Instead, according to Vähäniitty et al. (2002), the product roadmapping process starts with defining or revising, and then analysing the strategic mission and vision of the company. The purpose is to clarify and communicate the company’s area of business. This is because all companies should have a sufficiently clear idea of their purpose and desired future to be written down before their operations are planned in more detail. The company’s mission and vision acts as a guideline for shaping the product vision and choosing between strategic alternatives. Thereafter, major trends in the business environment, such as potential customers, competitors, the industry and developments in relevant technology, are observed and identified. The purpose is to create an understanding of the desired focus and position of the company and its products, as well as examine and guide the selection of technology (Vähäniitty et al., 2002).

According to Nuseibeh and Easterbrook (2000), capturing or eliciting of features is regarded as the first step in the requirements engineering process. When capturing features, gathered information often needs to be interpreted, analysed, modelled and validated to be sure that a sufficiently complete set of features of a product have been collected. Thus, capturing features closely relates to other roadmapping activities.

One of the most important activities in the capturing features phase is to find the problems that need to be solved, and hence identifying the product boundaries. The boundaries define, on a high level, where the final delivered product will fit, e.g. with

which target markets and which potential customers. Identifying and agreeing on the product's boundaries affect all the following feature-capturing activities. Therefore, the identification of stakeholders, user classes, goals, tasks, scenarios, etc. all depends on how boundaries are chosen. The identified stakeholders are persons or companies who stand to gain or lose from the success or failure of the product. Usually, the stakeholders include customers or clients, developers, and users, for instance. In addition, goals should be captured early in the product roadmapping process. The goals should denote the objectives a product must meet, and focus on the needs of the stakeholders (Nuseibeh and Easterbrook, 2000). Thereafter, the product features can be identified and gathered by communicating with all stakeholders (Parviainen et al., 2003).

There are several methods, which can be used during the feature capturing process. These methods include contextual inquiry, observation, prototyping, and scenarios (Parviainen et al., 2003). However, based on the literature, features are captured into product roadmaps by using, for example, group elicitation techniques. For instance, Nuseibeh and Easterbrook (2000) suggest that these techniques aim at improving the stakeholder agreement and buy-in, while exploiting the team dynamics to elicit a richer understanding of needs. According to Phaal et al. (2003a), the group elicitation technique includes a workshop or series of workshops which bring together a range of expertise, supporting the rapid capture, structuring and sharing of knowledge, together with simulating and brainstorming participation. During the workshop, all ideas for features in a product are collected in a roadmap template, which is also called product backlog or product feature document. The document provides a systematic way of collecting feature suggestions continuously from all participants and stakeholders (Phaal et al., 2003a; Rautiainen et al., 2003). There are also other techniques used in capturing features into roadmaps, such as market research, interviews, surveys and analysis (Phaal et al., 2003a). This is because product roadmaps require a good understanding of the markets and application in order to define the products in terms of customer needs (Groenveld, 2007).

In the capturing features phase, companies can collect features from several preferred sources for the product. Parviainen et al. (2003) propose the following sources as inputs for this phase: business requirements, customer requirements, user requirements, constraints, in-house ideas and standards. In addition to these perspectives, Albright and Kappel (2003) point out that the roadmapping team can define a market section which includes an analysis of competitors, a market research, and a product evaluation.

2.3.2. Analysing features

When the product features have been collected, the roadmapping team should begin to analyse the collected features. There are several methods that can be used during the feature analysis, for example knowledge-based critiquing (Fickas and Nagarajan, 1988), and feature-oriented approach to model requirement dependencies (Zhang et al., 2005). The purpose of the analysis is to remove uncertainty, identify and resolve conflicts as well as to analyse the feasibility of the gathered features, and make a resource and cost estimation (Soffer et al., 2005). Also, the purpose is to reveal dependencies between the requirements. Thus, the team should decide the methods required to evaluate the features.

Other issues to be considered in feature analysis are predictability of outcome, internal competencies in the organization, and opportunities for technology improvement (McCarthy, 2003). Also, the company's internal factors, such as human and financial resources, competencies and infrastructure, should be taken into account when analysing the features (Vähäniitty et al., 2002). Furthermore, there might be some factors restricting or improving the product features. For example, in the development of the mobile phone, the amount of the memory affects the features that are going to be included in the phone. This is because the memory might not be sufficient to implement all the presented features. Therefore, in the analysing phase, it should be analysed which features can be included in the phone and which have to be excluded. On the other hand, new technologies and development methods, unavailable earlier, can improve the product development; thus, these possibilities should also be analysed.

Thereafter, the gathered features and capabilities should be mapped into groups (Phaal et al., 2003a). According to Albright (2002), the features should be grouped by the product drivers that the features most strongly affect. Instead, according to Phaal et al. (2003b), the features should be first grouped, and then the groups should be arranged in terms of impact on the market and business drivers defined in the earlier phase.

The feature grouping is followed by feedback and discussion to identify synergies and gaps. The purpose of this practice is to ensure that all layers of the roadmap have been considered (Phaal et al., 2003a). If gaps are found, the team should take actions to close the gaps, e.g. by filling them with new product features. The gaps may also include a key technology that must be included in the product to meet a high-priority customer and market need. Hence, it should be estimated whether to develop or acquire the needed key technology (Albright and Kappel, 2003). Also, in this phase, the impact on market and the business drivers of the gathered product features are assessed, and alternative product strategies are considered (Phaal et al., 2003b; Holmes et al., 2004). Based on the feature analysis, the product vision is revised and captured as product roadmaps (Vähäniitty et al., 2002).

2.3.3. Prioritising features

Prioritising features is difficult and time-consuming, since features are related to each other. Therefore, it is complicated to schedule features based on priority only. Hence, at this point interdependencies between features should be explored, identified, and managed (Carlshamre et al., 2001). In product roadmapping, the product features should be prioritised so that the most important features are implemented first and the less important features are left until later, and the least important features are most likely omitted if the schedule or budget is insufficient (Greer and Ruhe, 2004).

According to Albright and Kappel (2003), the most important goal of roadmapping is to identify and focus strategy and product development on the few most important elements for success. Therefore, the roadmapping team should try to define the two or three most important drivers, elements or issues. That is, identifying the highest priorities. Albright (2002) points out that to achieve the main objectives, the team can define an action plan of a roadmap. The action plan identifies the highest priority features, and leads the team to schedule, budget, and staff them to accomplish the goals. With the action plan, the team can make sure that all feature gaps are closed.

There are different methods and tools for use during prioritisation. For instance, features can be prioritised by using informal and formal prioritisation methods. From the informal prioritisation method viewpoint, for example, Blotner (2004) suggests that initial feature prioritisation is done by the roadmapping team only using the identity information. This means that each feature is presented and input from all team members is gathered. Then the team attempts to agree on a spot for the product feature in the feature priority list. If consensus cannot be reached, the project manager either makes the final decision concerning the prioritisation or gathers enough information for the team to come to a consensus. Also, according to Phaal et al. (2003a), the roadmapping team conducts the prioritisation of the product features. The priori-

tisation of features is based on preparing an outline communication roadmap so that the priorities can be identified through feedback and discussion in a workshop session.

There are also theories that support the fact that formal feature prioritisation methods are used during product roadmapping. For instance, Phaal et al. (2003b) refer to Quality Function Deployment (QFD[1]) (e.g. Griffin and Hauser, 1993) as an often-used method for supporting product design in product roadmapping. Also, according to Groenveld (2007), the roadmapping process is quite frequently supported by methods such as QFD, because it is a customer-oriented approach that guides the roadmapping team at the beginning of the product roadmapping process. Typically, QFD is used because it is similar to roadmapping; they both require multidisciplinary communication and decision-making. QFD also helps to focus on the market requirements and translate these requirements into appropriate product characteristics, which facilitate feature prioritisation and, in consequence, product roadmapping. Groenveld (2007) also points out that there is one essential difference since roadmapping implies both current and future new features whereas QFD is generally restricted to available features. According to McCarthy (2003), the simplest way to use QFD is to develop a matrix in which the product needs are listed on the left side of the matrix and along the top are listed the features used to address these needs. The degree of alignment is rated pertaining to how well the feature meets the needs.

After prioritising the features, when the relative priority of each feature is established, product roadmap construction should start from defining the major and minor release cycles (Vähäniitty et al., 2002; Wiegers, 2003). The construction should be continued with defining the business features and expectations for the upcoming releases. When business features and their objectives are included in the feature repository and their history is traced, the rationale behind the roadmap evolution will become visible (Vähäniitty et al., 2002). With feature prioritisation, the construction of the product can be planned to provide the highest value at the lowest cost. However, even the low priority features should be documented, because their priority might change later and knowing them will help developers to plan future enhancements (Wiegers, 2003). Prioritisation is also important if features are dropped in case they cannot be finished in time for the release (Rautiainen et al., 2002).

2.3.4. Roadmap validation and agreement

In the roadmap validation and agreement phase, the planned product life cycle is estimated and the mix of development efforts is evaluated. The purpose is to check the financial rationale and assess whether the planned development is compliant with the product and company vision (Vähäniitty et al., 2002). Also, missing features and inconsistencies are discovered (Grynberg and Goldin, 2003). At the same time, the content of the roadmap and its key messages are considered, and the gathered data is validated with internal expert information (Wells et al., 2004). Thus, suggested by McCarthy (2003), the roadmapping team should review the product roadmaps to determine whether the goals of the roadmapping effort have been met. If modifications to the roadmaps are required, the roadmapping team should define a revised action plan.

There are different variations how product roadmaps can be validated. For instance, according to Phaal et al. (2003a), the roadmap validation happens by ensuring that all layers of the roadmap are considered and all necessary information is included. Instead, according to Nuseibeh and Easterbrook (2000), roadmap validation is conducted by identifying the most important goals of each participant and then ensuring that these goals are met in the roadmap.

Finally, all product releases within the scope of two to three years and their content, i.e. features, are agreed and possible disagreements among stakeholders are resolved. This phase calls for effective communication on the product releases and features between different stakeholders, especially if the stakeholders have different goals (Nuseibeh and Easterbrook, 2000; Grynberg and Goldin, 2003). Thereafter, when the strategic issues are identified, discussed, and actions agreed the product roadmapping process can be taken forward (Phaal et al., 2003a).

2.3.5. Change management of the roadmap

Change management is part of product development but it also affects roadmapping, since product roadmaps should evolve as the environment, in which the product operates, changes and stakeholder needs change. Thus, managing the changes is a fundamental activity in the product roadmapping process. In product roadmapping, changes are managed by using tools for configuration management and version control, and exploiting traceability links to monitor and control the impact of changes in different parts of the roadmap documentation (Nuseibeh and Easterbrook, 2000). Furthermore, Richey and Grinnell (2004) have suggested that maintaining the roadmaps could be supported with building a composite roadmap digitally. According to them, it is a fast and simple technique and it allows the owners of each portion of the roadmap to maintain control of edits and changes.

Typically, changes to product roadmaps include adding or deleting product features, and fixing errors. Features are added because stakeholder needs change or because they were missed in the initial analysis (Nuseibeh and Easterbrook, 2000). When new features are added to the roadmap, it means that some of the other features must be excluded (Rautiainen et al., 2003). Usually, features are deleted during development to prevent cost and schedule overruns (Nuseibeh and Easterbrook, 2000). Fixing errors and improvement suggestions are included in the product roadmap and planned into future product releases. These re-prioritisations of the product features are the responsibility of the product roadmapping team. Further, the priorities are reformed in different sessions with internal experts and selected customers and partners (Rautiainen et al., 2003).

The requirement change management process defined by Pozgaj et al. (2003) constitutes the following phases: feature change identification, analysis of the change, definition of the change impact, definition of the change actions, decisions, and implementation of the feature change. The roadmapping team members are responsible for continuously tracking and supervising features in order to discover features' changes. When the indications for feature change occur, the features should be carefully analysed and the feature change identified. After the change identification, all consequences of the change must be analysed (Pozgaj et al., 2003). Each proposed change should be evaluated in terms of the existing features and architecture so that the trade-off between the cost and benefit of making a change can be assessed (Nuseibeh and Easterbrook, 2000). The feature change analysis should focus on the impact of the feature change on other features and on the definition of the features' change actions (Pozgaj et al., 2003). The purpose of the feature impact analysis is to identify what to modify to accomplish a change, or to identify the potential consequence of a change (Arnold and Bohner, 1993). Traceability links help to scope a possible impact of change and to define which parts are related to which other parts according to specific relationships (Arnold and Bohner, 1993; Nuseibeh and Easterbrook, 2000). The impact analysis is important, since product features are frequently interdependent. Thus, a small change could create a major impact because of the many ripple effects (Ebert and Smouts, 2003). The feature change

[1] Drs. Yoji Akao and Shigeru Mizuno originally developed QFD in the early 1960s. They also co-founded the Quality Function Deployment Institute. http://www.qfdi.org/.

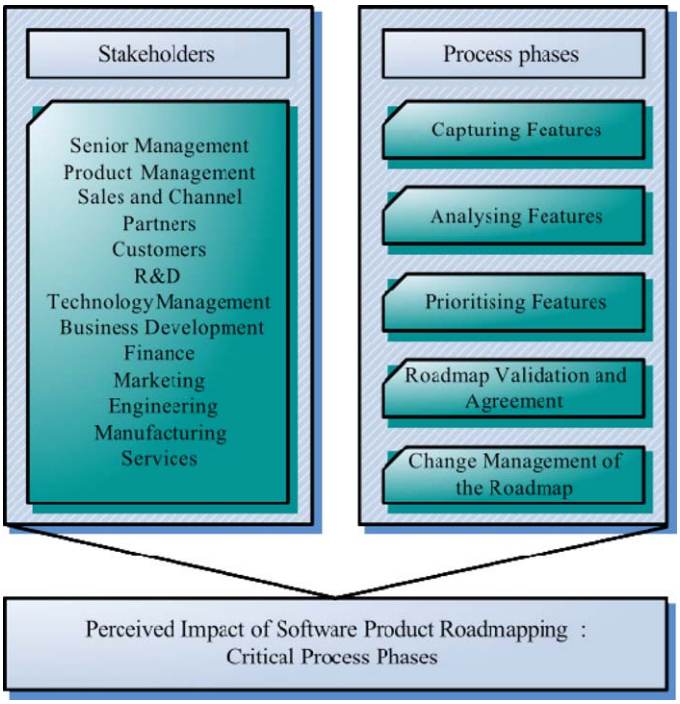


Fig. 1. Research framework for software product roadmapping.

actions should be defined for all development areas that are under the impact of feature change to enable immediate response to the feature change. The purpose is to provide sufficient information for the decision about the realization of feature change (Pozgaj et al., 2003).

Finally, a Change Control Board (CCB) makes all the decisions concerning the change. The CCB can be the same as the roadmapping team or the company can have a separate group of people forming the CCB. On the other hand, the CCB can also be a part of the roadmapping team. In that case, only the most important representatives, e.g. the owner, participate in the decision-making. The CCB decides on each feature change request according to the information provided by the feature change analysis to implement or to refuse the feature change (Pozgaj et al., 2003). Thereafter, the final phase of the feature change process is the implementation, if the feature change is to be carried out.

2.4. Research framework

A conceptual research framework is used in this study for theory building. As presented by Miles and Huberman (1994), a conceptual research framework describes in graphical or narrative form the main aspects to be studied meaning the key factors, constructs or variables, as well as the relationships among them. They explain that theory-building relies on a few general constructs that contain a set of particulars. Elements are labels of the theory which are put in intellectual "bins" including many distinct events and behaviours. Bins are formed based on theory and experience as well as from the general objectives of the study. A conceptual framework is about setting out bins, naming them, and gaining clarity about their interrelationships (Miles and Huberman, 1994).

The research framework for software product roadmapping is created based on the existing literature. The purpose of the research framework is to help to structure and present the research results of this study. The research framework for software product roadmapping is presented in Fig. 1.

The framework aims at providing answers to questions that relate to the following elements in product roadmapping: stakeholders and process phases. The research framework also includes the perceived impacts of software product roadmapping, i.e., the main benefits and challenges faced during product roadmapping and the most critical phases of the process. The empirical results in Section 4 are presented in accordance with the research framework at hand.

It should be acknowledged, that there are also other frameworks presented in the literature that include product roadmapping, for example the reference framework for software product management presented by van de Weerd et al. (2006). However, their framework has broader perspective to the software product management than we do in our framework. Among other activities, their framework includes portfolio management, and we focus purely on the content of the product roadmapping activities and stakeholders. According to van de Weerd et al. (2006) the product roadmapping includes activities such as theme and core asset identification, and roadmap construction. These activities are included in our framework in to the product roadmapping phases. For example, theme and core asset identification are included in the capturing features phase. In our framework, the roadmap construction is not seen as separate activity of the process; instead, the product roadmap is created and revised through out the process.

3. Research design

In this section, the research methods and goals as well as research context and data collection are defined. First, research methods and goals (Section 3.1) are presented and then, the research context and data collection (Section 3.2) are described.

3.1. Research methods and goals

The research is a multiple-case study. The purpose of the multiple-case study is used either when the results from the earlier case study are verified, i.e. similar results are predicted, or when contrasting results are obtained, but for predictable reasons (Yin, 1994). This research approach was used, since the theory on the product roadmapping was not yet well formulated. Therefore, this study was started by conducting a literature review (Section 2). Its purpose was to understand the current state and to find the gaps in the literature. As a result, a conceptual framework was created (defined in Section 2.4). It contains the essential elements of product roadmapping and was used in designing the empirical part of this study.

The empirical research was carried out as a qualitative research conducted with a questionnaire-based survey study (Oppenheim, 1992) and semi-structured interviews (Järvinen, 2001). In the questionnaire-based survey method, Alreck and Settle (1995) classify three methods of data collection: personal interviewing, telephone interviewing and mail data collection, of which all were used during the research. In selecting the respondent organizations for the questionnaire survey and interviewees, a purposive sampling (Nardi, 2003) was used. Purposive sampling involves designating group of people for selection because they have some traits that are important for the study (Nardi, 2003). In this research, the questionnaire respondents were selected based on VTT's (VTT, Technical Research Centre of Finland, 2009) electronic mailing lists. Also, the questionnaire was sent to European-wide Merlin research projects' partner companies (MERLIN ITEA project 2004–2007), since they were assumed to have knowledge about the field of study. The interviewees were, then, selected based on the questionnaire respondent's experience in product roadmapping. The emphasis of the empirical research was put on the interviews. The questionnaire was used to obtain data about product roadmapping and to obtain the right persons to be interviewed.

Table 3
Empirical research design.

Research method	Research questions relating to each element		
	Stakeholders	Process phases	Critical process phases
Questionnaire	Who should participate in product roadmapping? How many persons participate in the roadmapping process in your company?	How product requirements are captured into roadmaps? How requirements are prioritised during product roadmapping?	What is the most important phase in the roadmapping process? What is the most difficult phase in the roadmapping process?
Interviews	What roles does the roadmapping process include? Who participates into the roadmapping process and to which activities in it?	How does the roadmapping process begin? What are the main phases of roadmapping process? How each product roadmapping phase is conducted?	What are the benefits of product roadmapping? What are the problems of product roadmapping?

The content of both the questionnaire and the interviews were designed according to the research framework for software product roadmapping, presented in Section 2.4. The questions were divided along the framework into three elements: stakeholders, process phases, and critical process phases. An example questions from both research methods along with the elements are presented in Table 3.

The goal of this study is to increase the current empirical evidence on product roadmapping from three elements: (1) the stakeholders of the product roadmapping (i.e. who the main stakeholders are and what kind of role they have during the product roadmapping process), (2) the product roadmapping process phases (i.e. what are the main process phases and how they are conducted), and (3) the benefits and challenges and critical points of product roadmapping. In addition, a tentative research framework for software product roadmapping is proposed (Section 2.4) based on the current literature. The research framework is for designing the research as well as ordering and presenting research results.

3.2. Research context and data collection

In this research, the empirical study was carried out in five phases: (1) formulating the questionnaire, (2) pilot testing of the questionnaire, (3) analysis of the questionnaire results, (4) interviews, and (5) analysis of findings. Fig. 2 illustrates the process of the conducted empirical study.

Firstly, a literature research was conducted in order to understand the state-of-the-art of product roadmapping. In addition, the purpose of the literature review was to focus research on a few of the most important issues from the field of research, and to prepare the proposed questionnaire based on the literature review. The data collection was carried out, firstly, in the form of a questionnaire study to collect background knowledge of the case companies on the application of product roadmapping. The interviews, then, were conducted to collect more in-depth knowledge about product roadmapping in different companies.

The questionnaire was built in a structured form to provide the respondent with specific alternatives for each question. However, in order to get more in-depth information, a respondent could also reply with an open answer to each of the questions. Thus, the questionnaire included both structured and unstructured questions. The structured questions were selected to attain a brief response time and therefore to receive more replies. Also, the structured questions were chosen to make analysis easier and conclusions stronger. The questions for the inquiry were planned and arranged carefully in advance to attain the right form of questions and to avoid misunderstandings. Thus, the questionnaire was pre-tested inside VTT (VTT, Technical Research Centre of Finland, 2009) among researchers, and based on the pre-test results, small modifications to the questionnaire were made.

After the pilot testing of the questionnaire, in summer 2006, the questionnaire was sent to potentially interested contacts by using purposive sampling (Nardi, 2003), i.e., to companies assumed to have experience and interest in product roadmapping. An electronic mailing list was prepared and the questionnaire was e-mailed to over 600 respondents. The high receiver rate can be partly explained by the fact the mailing list was not ranked, e.g. according to respondents' roles, thus the questionnaire was also sent to those respondents that were not actually involved with the scope of research. As a result, a total of 59 responses were received, seven of which lacked experience in product roadmapping and were hence unable to fill the form. Though the resulting response rate of the questionnaire was quite low (10%), the 52 replies from 34 different companies can be claimed to provide a wide enough perspective for further analysis, especially for selection of persons to be interviewed. One of the replies, however, was excluded from the analysis due to an incompletely filled questionnaire form.

The respondent companies originated from Finland, Sweden and the Netherlands. All companies were involved in software product or service development in the field of information and communications technology (ICT) industry. Their scopes varied from own product development to the development of components for external partners. The size of the case companies, measured as the number of overall employees, was distributed among the given categories in the questionnaire (under 10 employees, 10–49 employees, 50–250 employees, and over 250 employees) with an emphasis on medium and large companies. The distribution of answers according to company size is illustrated in Fig. 3.

After the questionnaire study, the interviews were planned. The purpose was to gain in-depth knowledge on how product roadmapping was conducted in case companies. The interviews were semi-structured, and as Järvinen (2001) describes, the semi-structured interviews include structured questions and proceed along certain vital themes of the research. The vital themes of research were selected based on the literature analysis, presented

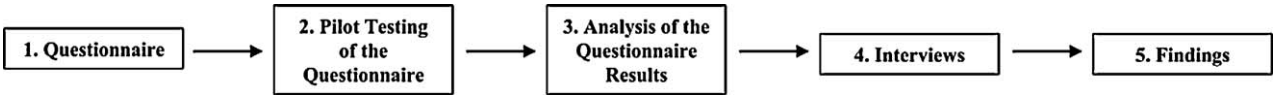


Fig. 2. Empirical research process.

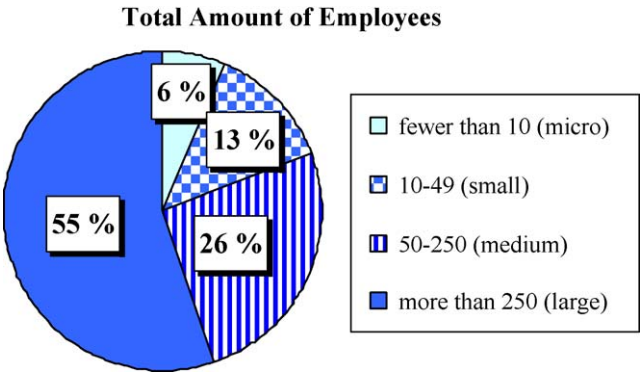


Fig. 3. Size of the case companies.

in Fig. 1, and findings of the questionnaire studies. As is characteristic of semi-structured interviews, the themes were the same for all the interviewees, but the questions varied between the different interview sessions. Equally, the interview questions were partly planned in advance, but not in a detail formed or arranged. Moreover, the intention in the interviews was to emphasize the interviewee's experiences and their own opinions on the field of study. Furthermore, the interviews could be considered focused interviews, since the respondents were interviewed personally for a short period of time, i.e. not more than an hour (Yin, 1994).

Altogether, nine persons representing eight different companies were selected to be interviewed. Two of the interviews were face-to-face, and the other seven were conducted by phone. All the interviews were recorded and transcribed. Table 4 provides a summary of the conducted interviews indicating company nationality, company size, and the role of the interviewee.

4. Results

In this section, the research evidence from the empirical study is presented and aligned with the research framework developed in Section 2.4. Firstly, in Section 4.1, the empirical evidence to identify the main identified stakeholders of the product roadmapping process is defined. In Section 4.2, the findings related to the roadmapping process and its suggested phases are presented. In Section 4.3, the perceived impact of software product roadmapping is disclosed meaning e.g. the research evidence revealing the critical phases in product roadmapping and the main benefits and challenges of product roadmapping.

4.1. Stakeholders of the product roadmapping process

In order to clarify different definitions in the literature about the stakeholders of the roadmapping process, the questionnaire respondents were asked to select different functions of the organization that should take part in the product roadmapping process. Also, the purpose was to find out the most important stakeholders. The stakeholders were divided into the following categories: product management, finance, engineering, marketing, manufacturing, services, development, customer and partner representatives, and other. Fig. 4 illustrates the questionnaire replies relating to the stakeholders of the roadmapping process.

According to the respondents, the following were considered to be the most important stakeholders of the product roadmapping process: product management, marketing, customer and partner representatives, and development including manufacturing and engineering. From these stakeholders the customer and partner representatives were the only external stakeholders to be involved in product roadmapping based on the questionnaire results. However, all the categories received a number of responses and, thus, can be considered somehow important. Additionally, 21% of the respondents thought that other groups of representatives were also needed during the process as well. These groups included various management stakeholders, e.g., top management, senior management, and human resource management, as well as final end-users and sales personnel.

When several different functions from organization(s) participate in the roadmapping process, the number of participants can become quite large. Therefore, the questionnaire respondents were asked to describe how many persons typically participate in the roadmapping process in their company from the given categories (1–5 persons, 6–10 persons, 11–20 persons, 21–30 persons, and other). In total, 50 responses were received for this question.

It may be considered surprising that the category "1–5 persons" was selected by 34% of the respondents, even though most of the companies were considered large. Thereafter, the second highest rate was given to the category "6–10 persons" with 30% of the replies. The third largest category was "Other" with 16% of the replies, in which the respondents explained that more than thirty, fifty, or hundred persons participate in the roadmapping process. Additionally, some of the respondents described that hundreds or several hundreds, or more than two hundred persons take part in the roadmapping process. The two last categories were not very common, since the category "11–20 persons" received 14% and the category "21–30 persons" received only 6% of the total number of replies.

The questionnaire data also implicated that the company's size and the number of participants in the roadmapping process clearly correlate. According to the majority of the small companies with fewer than 50 employees, one to five persons participate in the product roadmapping process. Instead, in medium sized companies with employees from 50 to 250, the number of participants in the roadmapping process was from six to ten. Furthermore, in case of large companies with more than 250 employees the number of participants was more than 30, and could be even several hundreds, which made the process very complex, as one of the respondents explained.

The interviewees emphasized that not all the participants were necessarily present during each phase of the process. Instead, the participants only participated in those phases that affected their

Table 4
Summary of interviewee profiles.

Interviewee	Company nationality	Company	Company size (employees)	Role of the interviewee
1	Finnish	A	>250	Manager
2	Finnish	A	>250	Group manager
3	Finnish	B	50–250	Chief technology officer (CTO)
4	Finnish	C	>250	Product planner
5	Finnish	D	>10	Program director
6	Finnish	E	>250	Group manager
7	Finnish	F	>250	Chief engineer
8	Swedish	G	>250	Senior researcher
9	Finnish	H	50–250	Manager

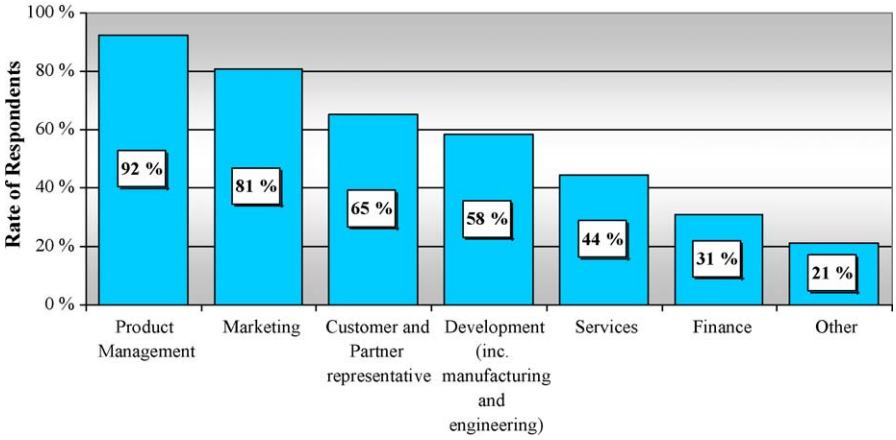


Fig. 4. Stakeholders of the product roadmapping process.

work or knowledge. Especially, when the company was large, there were different types of roles from different kinds of companies that were involved in different phases of roadmapping. Thus, different meetings were reported to be held with different groups of participants to focus on specific areas of the product. One of the interviewees pointed out:

> "Separate feedback or prioritisation sessions should be held, and joint meeting are held to discuss all issues." [Manager, company H]

Instead, in smaller companies when the process included fewer participants, all the participants could be present in each phase and meeting of the process.

According to the interviews, the product roadmapping process participants had only two basic roles: as a member of the product roadmapping team and as a product or solution owner. In addition, there was a third role mentioned in the literature, the facilitator's role, which was not considered to be important by the interviewees. According to the interviews, the most important role was considered the owner's role as this person had the overall idea of the desired roadmap, and there should be someone in possession of the roadmap. The role of the owner was also considered to include the collection of input, holding the roadmap together, making the needed changes to the roadmaps, and taking care of the information flow both within and outside the company. Instead, the stakeholders of the roadmapping team were to bring input from different viewpoints to the roadmap, e.g. to schedules and to product features. The team was also responsible for evaluating and prioritising features as well as reviewing created roadmaps.

4.2. Product roadmapping process

Among the interviews, product roadmapping was identified as a continuous process where the roadmapping team has meetings biweekly, quarterly, or biannually, in which roadmaps are updated and reviewed. The majority of the interviewees replied that the product roadmapping process begins with customer requirements. These requirements could be, for instance, proposals for improvement or new product features as well as the customer's goals or expectations. The requirements could also come from the company's internal research unit or through competitor analysis. In some cases, changes in the standards were reported to cause the trigger for a product roadmapping. The product roadmapping process could also begin with:

> *"By defining what product features were emphasised in the markets." [Group Manager, company A]*

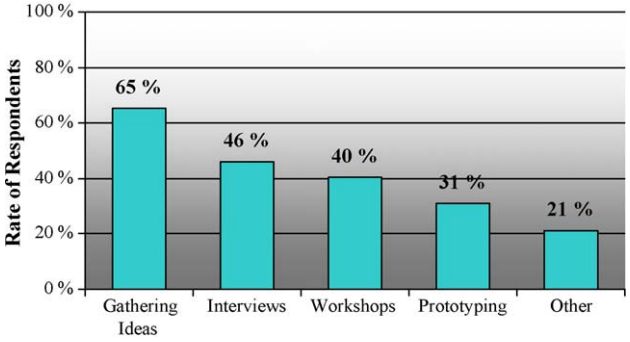


Fig. 5. Methods for capturing features.

The product roadmapping process consisted of three to six phases according to the interviews. Although the number of phases varied, in almost all the companies the same tasks could still be identified as elements of the product roadmapping process.

4.2.1. Capturing features

As mentioned in the literature, there are several methods, which can be used during capturing features. To find out what kind of feature capturing methods were used in the case companies, the questionnaire respondents were asked to select the used method(s) from the given categories (with prototyping, with interviewing, gathering ideas over time, in some kind of workshops, and other). All 52 responses were included in the analysis; most of the respondents selected more than one alternative.

Although the literature in the field of the study recommends using workshops for capturing features, the questionnaire replies revealed that some other procedures were used more often than workshops. Accordingly, the most commonly used method was gathering ideas over time with 65% of the replies and after that, interviews with 46% of the replies. Only 40% of the respondents used workshops to gather features, and 31% of the respondents used prototyping. Additionally, 21% of the respondents used some other methods to capture features. These methods included, for instance, market and technical research, following market development and standards, as well as analysing the distribution channels and their requirements. The most commonly used methods for capturing features are presented in Fig. 5.

According to the interviews, during capturing features, the features originated from several sources of information such as from current market trends and standards. Thus, knowledge of different participants was combined. The stakeholders of capturing features were typically the collaborators, sales, management, or product

architects, as well as from a company's own research, for example, through customer and competitor analysis. In addition, one of the interviewee's replied:

> *"Features are typically captured and feedback is collected through trade shows and by distributing a free evaluation version through the internet". [CTO, company B]*

In some case companies, the idea of the product was created together with collaboration partners, e.g. through brainstorming. In the other case companies, possible participants and co-operation partners were clarified after a product idea was created inside the company. However, as one of the interviewees noted:

> *"Capturing features has two phases: someone has an idea for a product feature and another one writes down what the feature actually means. Based on that information it can be verified that everyone has understood the idea of the product feature correctly." [Chief Engineer, company F]*

4.2.2. Analysing features

Various methods were used for analysing features in the case companies based on the interviews. Some of the case companies used domain-specific knowledge and experience as well as interviewed experts during the analysis phase. Further, in one case company, the major features (i.e. features requiring more effort to be implemented) were analysed through a feasibility study, as suggested in the literature. The feasibility study included both a technical and cost analysis. Based on the analysis estimations for the revenues, sales and implementation could be made. Minor features were not analysed, but rather were further defined in order to make the decision whether they fit into the product content or not. In the latter case, they were dropped or postponed as candidates for the next product release. Usually, the analysis of features was conducted by the experts from different stakeholder groups, e.g. from the viewpoint of sales, technical or strategic value. However, in one case company:

> *"Analysing features is the responsibility solely of the product manager or person in charge of the product." [Product Planner, company C]*

Typically, the three most important factors in analysing features were identified as: (1) the estimated cost, (2) technical requirements (e.g., specific hardware needs), and (3) the central use cases (e.g. when and how the functionality is being used, and what else should function at the same time).

In more detail, analysing features was found to consist of three phases. Firstly, it was verified whether the features were understood correctly, for example, whether the features were recorded exactly, and if the feature description included information concerning the feature's functionality and limitations. Secondly, it was figured out what kind of technology would support the implementation of the features. This was typically done by the architects and technical persons and could involve prototyping or sketching the software architecture. This was reported to help in understanding the implementation of the features from a technical viewpoint. Thirdly, after creating basic knowledge and understanding, it was estimated how much it would cost in terms of time, money, and external recourses to implement the features.

4.2.3. Prioritising features

According to the literature, the product features can be prioritised both with formal or informal methods. To find out what kind of prioritisation methods were typically used, the questionnaire respondents were asked to select either formal, informal or both methods. The clear majority of the 52 respondents selected informal methods with 69% of the total amount of replies. Thereafter, 23% of the respondents used formal methods only, and 8% of the respondents used both methods for prioritising features. The formal prioritisation methods included Analytic Hierarchy Process (AHP) (Saaty, 1986), Quality Function Deployment (QFD) (Griffin and Hauser, 1993), EVOLVE (Greer and Ruhe, 2004), and Distributed Prioritisation (Regnell et al., 2001). Additionally, in one case company, formal methods contained business cases, probability analysis (win/lose), and customer and market importance balancing.

Moreover, two of the interviewees had experience on the use of formal prioritisation methods; Distributed Prioritisation and AHP. Distributed Prioritisation was used:

> *". . .when it was supported with tools allowing simulation of multiple approaches and weighting of answers." [Manager, company H]*

It was also used to support distributed knowledge, since within a global company it was important to gather distributed priorities from all perspectives. Instead, AHP was used to find out focal points, for instance. Accordingly, also AHP was reported to support distribution. That was partly because the case company's user interface supported sharing and distributing information between companies.

According to the interviews, both functional and non-functional requirements were prioritised at the same time. In addition, there were no special methods for prioritising non-functional requirements. Prioritisation was typically informal, and the informal prioritising methods varied among the case companies. For instance, the priorities could be created by using a calculation system. In that case, each feature was given a point, which could also be a weighting factor, work contribution, or incurring costs. After giving the points, it could be seen which of the features had won. Based on the points, the order of priority was formed. Instead, in some case companies, the number of customerships per feature or values related to technical importance, market value, or ease with return on investment (ROI), guided prioritisation. On the other hand, in some cases, more information relating to the features was needed to be considered during the prioritisation. For example, customer preferences, legislative or release specific features, and real world matters guided prioritisation.

Typically, the order of priority of the features was a result of a collective decision in the roadmapping team. In some cases, the decision could be made also by the product manager, CTO, the owner, or the person in charge of the product. Usually, minor problems concerning the priorities were handled inside the roadmapping team and the major problems were resolved by the management team of the company or companies. Additionally, in some case companies, the customer made the final decision concerning the priorities, especially, when intermediate versions from the product were important to the customer.

4.2.4. Roadmap validation and agreement

Commonly, there were two means for performing validation according to the interviews. The roadmaps were either validated through reviews or through the unit's business improvement and customer feedback.

According to the first group of interviewees, roadmaps were validated through negotiations, meetings, or reviews. Accordingly, reviews were considered the most efficient way of performing validation, especially when roadmaps were reviewed with an adequate number of persons. During the review process, inputs as well as comments were collected, and if there was any essential new information, the necessary changes to the roadmaps were made. Hence,

in order to widely spread and review the roadmaps:

"Roadmaps are kept in either paper or electronic format." [Program Director, company D]

Additionally, the roadmap validation could be manifested in contracts. It could even be a legal contract, if financial matters were involved. Moreover, when roadmaps were created together with collaborators, validation of the roadmap was also performed in mutual meetings with the partner. Typically, the customer confirmed that the roadmap was good.

Instead, according to the other group of interviewees, validation took place when the customers started to buy or not to buy the product. Thus, validation came through the unit's business improvement and customer feedback. If negative results were obtained they were analysed, for instance to reveal why product development had gone in an unprofitable direction or why the newest version did not answer to the customer's needs. Roadmap validation could also come through commercial success, e.g. as a function of the number of sold products.

The roadmap agreement was often made in a meeting together with the roadmapping team. The agreement could also be made by the product owners, the product managers, or CTO, depending on the case company, and if needed, the management participated in the meetings as well. In such a meeting, the product roadmapping team was reported to make a mutual decision about the roadmap from which point onwards the development work would then proceed. The roadmap agreement was made in order to have a mutual understanding about the product that was being developed. The agreement also made the roadmap official. Thus, with the agreement, it could be ensured that commitment existed and everybody knew the decided matters.

4.2.5. Change management of the roadmap

According to the interviews, changes to the roadmaps came from delays in the product implementation as well as when new or unnecessary product features were discovered during product development. Additionally, when partners or customers were informed about the new solutions, they typically brought up matters that might not have been taken into account earlier. These new matters had to be either brought out with the ongoing releases or left at the roadmap stage for forthcoming product releases. The case companies reported having meeting practices for change management, in which change requests were handled and decisions concerning the change were made. One of the interviewees highlighted:

"We try to manage changes that affect schedules or money in joint meeting, from which meeting memos are created." [Chief Engineer, company F]

Thus, it was found important to record the meetings, e.g. create notes or meeting memos, in case there would be a need to verify the change decisions later on.

The roadmap change process typically went as follows. First, either one of the collaboration partners noticed the change request. Secondly, the effects of the change were analysed, i.e. an impact analysis was conducted. Third, changes were approved together with the partners, or a customer was requested to approve the changes. That was because all the decisions had to be conscious and those that were mostly affected by the change had to be able to affect the change decisions. Thereafter, it was verified that everyone had understood the changes. Finally, the changes were added to the rest of the features, i.e., a new roadmap or updated version of a roadmap was created.

The decision-maker of the change was reported to depend on the importance of the matter to be changed. Minor changes could be made by the product manager or CTO. Major changes were managed by the roadmapping team. Major changes were largely related to schedules, deleting important or key features from the roadmap, or adding bigger features to the roadmap, etc. Thus, these changes had to be communicated and approved by the same forum that had approved the roadmap. Instead, when the customer was financing the product, then the final decision was made by the customer. Moreover, in other collaboration situations, the first, preliminary decision concerning the change was made inside the company. Thereafter, the change decision was negotiated together with the collaboration partners in meetings, and they tried to reach an agreement. When unsolvable problems occurred during meetings, then management of the companies made the final decisions concerning the change. Therefore, in collaboration, the changes could not be unilaterally decided, instead they had to be approved mutually. Usually, the rules concerning the change management were known and approved before cooperation, for example, in the cooperation agreement.

4.3. Perceived impact of software product roadmapping

The interviewees had convergent ideas about product roadmapping. Based on the interviews, a roadmap is a plan about the company's future directions; in other words, it is a leading map where the company is going with its products. Thus, it is the means of structuring and arranging the product development, in order to know how to use certain resources. A roadmap holds the product development together by guiding what is to be done and when. Hence, product development is somehow deterministic and enables steering of the product implementation. A roadmap also gives a clear focus in the product development, and provides high-level understanding of scoping the strategy. On the other hand, one of the interviewees pointed out:

"Clear strategy allows better planning and commitment to the set plans." [Manager, company H]

Product roadmapping also improves predictability, and reduces surprises during product development. With a roadmap, tasks to be done can be prioritised, and thus resources can be allocated to the most profitable projects. It can even be verified from the roadmap that the right things are done at the right time. Additionally, with a good roadmap the customers' needs can be met with a product that they really want. Hence, roadmaps offer a competitive advantage. Moreover, a roadmap is a central tool for communication, and therefore it should be shown to the company's own staff and to partners. It gives a clear idea what is about to be done and enables communication about forthcoming strategic projects.

4.3.1. Importance of the phases

The questionnaire respondents were asked to rank the roadmapping phases from the most important to the least important (1 = most important and 5 = least important). The phases to be ranked were divided into the following categories: capturing features into roadmaps, analysing features, prioritising features, roadmap validation and agreement, change management of the roadmap, and other. Two of the replies were left outside the analysis since several "most important" alternatives were selected. Hence, the total number of replies was 50. It should be noticed that not all of the respondents ranked all the categories. Instead, some of the respondents only selected one to three phases that were considered the most important, and left the other phases outside the ranking. Thus, some of the categories have fewer replies. Fig. 6 illustrates the phases of the product roadmapping process and the ranking order given by the respondents.

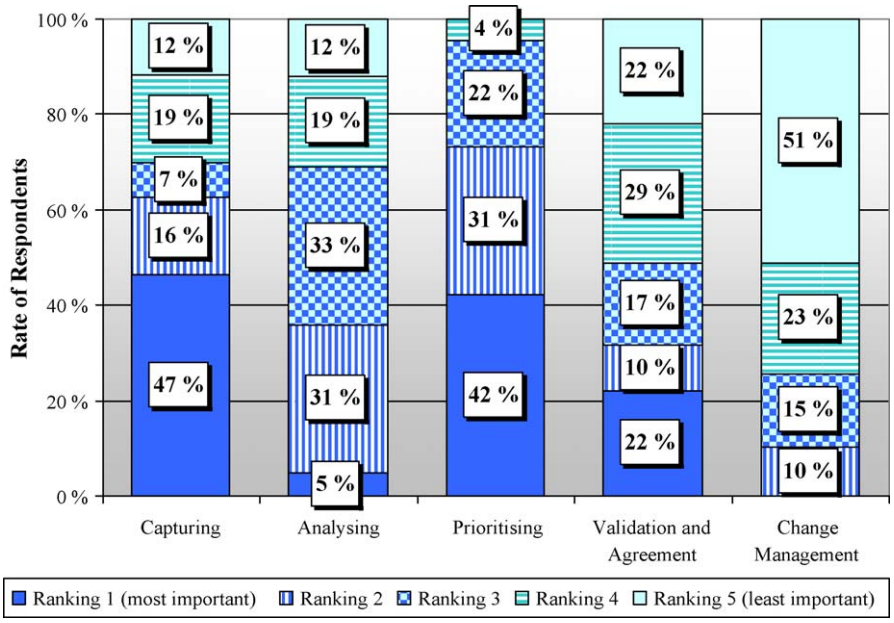


Fig. 6. The most important phases in product roadmapping.

The majority of the most important replies, 47% altogether, were given to the capturing features phase. After this came the prioritising features phase with 42% of the most important replies. Since none of the respondents thought that prioritising features was the least important phase in roadmapping, the prioritising product features was considered the most important phase in the product roadmapping, and then the capturing features. The roadmap validation and agreement was considered the third most important, and the analysing features phase was the fourth most important. The change management of the roadmap was not considered so important, since none of the respondents ranked this phase as the most important. Instead, the majority of replies gave this phase the rating of least important. This can be explained by the fact that the change management will not help, if all the other phases have gone wrong.

4.3.2. Difficulties of the phases

Thereafter, the respondents were asked to select one of the product roadmapping phases that was the most difficult. According to the clear majority of replies, the most difficult phase was prioritising features with 46% of the replies. The other phases were distributed quite equally among the replies; capturing features with 19% of the replies, validation and agreement with 17% of the replies, analysing features with 10% of the replies, and change management of the roadmap with 8% of the replies.

In order to find whether the company's size affects the most difficult phase in roadmapping, the companies were divided into the three size groups (small: fewer than 49 employees, medium: 50–250 employees, and large: over 250 employees). Based on the replies, both the medium and the large sized company groups thought that the most difficult phase of roadmapping was the prioritising features phase. Instead, according to small companies, the most difficult phase in the product roadmapping was the roadmap validation and agreement phase. Additionally, large companies faced difficulties during all the phases of roadmapping unlike the small and medium sized companies that faced difficulties only during prioritising features, roadmap validation and agreement, and capturing features into product roadmaps.

Furthermore, it was analysed whether the lifetime of the company's products had an effect on the most difficult phases of the roadmapping. The product service lives in use by customers were divided into the following categories: a lifetime from one to six years, a lifetime over six years, and another that included products with varying lifespans from one to more than 10 years. The companies with a product lifetime between one and six years had difficulties during all the phases of roadmapping except during the change management phase. Additionally, the companies with a product lifetime over six years did not face difficulties during the roadmap validation and agreement. Instead, companies that selected the category "Other" faced only difficulties during the roadmap validation and agreement, and the capturing features phase.

According to the interviews, there are problems relating to almost every phase of the product roadmapping process. Most commonly, the problems seem to relate to prioritising the features, managing changes, and maintaining roadmaps. Also, sharing information, communication, and making the roadmap agreement were considered difficult.

Collecting input was considered difficult, and more precisely, getting the right information and accurate knowledge was problematic. Also, background research and finding out both competitors' and customers' opinions was difficult. Hence, uncertain predictions about the future were attempted to be avoided by collecting feedback on the prediction accuracy. Additionally, it was highlighted that at the beginning of the product, the roadmaps could be more accurate which meant that a more precise product design could be created right from the beginning of the process.

In prioritisation, problems were caused by uncertainty about which features should be taken into which product version. It was also described that the strongest opinion may not offer the best understanding of the values, and thus prioritisation should be done carefully. Relating to prioritisation one interviewee revealed:

> *"If not done with clear vision and with strategic thinking it will lead to changes in plans and results in frustration very ineffective product release cycles." [Manager, company H]*

Also, prioritising customer requirements was considered a continuous problem and a challenge in the roadmapping process. Making an agreement was also considered difficult, since combining and processing different viewpoints was complex. Especially, consolidation of different wishes, needs, and technical implementation possibilities in an economical frame was considered

hard. Moreover, managing and maintaining product roadmaps was thought difficult. The problems were caused by a large number of changing matters, and the more precise the roadmaps were the harder they were to maintain.

Communication about features across organizational borders to ensure that everyone involved understands the meaning of the features was considered difficult. Additionally, communication in large companies was considered problematic. Thus, it was suggested that sharing information and its visibility should be improved. Moreover, it was feared that information would get into the wrong hands, since it was problematic to verify persons who should know about the roadmap, and that they were well aware of the content of the roadmap. Thus, it was a matter of concern, how to get the information without the danger of data leakage. Hence, the purpose was to prevent the information from the roadmap from being passed on to competitors.

5. Discussion

In this section, the research results are discussed. The literature review findings are compared against the empirical research results. In Section 5.1, the empirical findings from the product roadmapping stakeholders are compared against the literature findings presented in Table 1 and, in Section 5.2, the empirical findings from the product roadmapping process are compared against the literature findings presented in Table 2. Also, at the end of this section, common practices in product roadmapping are summarized. In Section 5.3, the critical process phases of product roadmapping are discussed.

5.1. Product roadmapping stakeholders

Based on the empirical research findings, the product roadmapping process only seemed to involve two main actors: the roadmap owner and members of the roadmapping team. Empirical evidence did not support the use of a facilitator, as proposed in the literature (e.g. Phaal et al., 2003a; Wells et al., 2004). Furthermore, in the literature (e.g. McCarthy, 2003), it is suggested that the roadmapping team is, among others, responsible for creating the roadmaps. However, the research results indicate that the roadmaps are created by the owner, while the main task of the team is to give input to the roadmap.

As presented in Table 1 several different stakeholders were mentioned in the literature to participate in the roadmapping process, the empirical findings exposed the following main actors: product management, marketing, customer and partner representatives, and development including manufacturing and engineering. Therefore, for example, participants from finance were not seen as important as suggested in the literature (Phaal et al., 2000a,b, e.g. McCarthy, 2003; Phaal et al., 2003a). The customer and partner representatives were the only external stakeholders to be involved in product roadmapping. Instead, according to van de Weerd et al. (2010) the external stakeholders include representatives from the market, partners, and customers. Additionally, based on the empirical findings, the number of participants in the roadmapping process was directly connected to the size of the company. In larger companies, more stakeholders participate in the process. However, not all the participants were necessarily present at each phase of the process. Instead, they only participated in the phases that affected their work or where their particular knowledge was needed or relevant.

It was suggested in the literature (presented in Table 2), that the roadmapping process is different in every company (Phaal et al., 2004; Groenveld, 2007). However, the empirical evidence suggests that the contents of the roadmapping processes are still almost the same. The tasks to be done during the roadmapping process were the same in spite of the fact that the case companies had identified a different number of phases and different names for the phases. As a result, the following phases of a common product roadmapping process are proposed in this paper, i.e., capturing features, analysing features, prioritising features, roadmap validation and agreement, and change management of the roadmap.

5.2. Product roadmapping process

According to the empirical findings, the product roadmapping process was considered a continuous process that was a part of the product development process. Unlike what is suggested in the literature (Vähäniitty et al., 2002) that roadmapping process starts with defining or revising, and then analysing the strategic mission and vision of the company, the results of this research propose that the product roadmapping process seems to begin with the customer requirements. In some cases, the process also begins from product requirements emerging from the company's internal research, or through competitor, market or standard analysis.

It was presented in the literature (e.g. Phaal et al., 2003a) that workshops were most commonly used for capturing features. Instead, the questionnaire study revealed that gathering ideas over time was the most commonly used method for capturing features and interviews after that. Furthermore, the features could come from several sources of information, e.g. market trends and standards. Combining knowledge from different participants was seen important, and therefore several stakeholders were advised to be involved in capturing features, e.g. collaborators, sales, management, product architects, and market researchers.

The methods for analysing features were almost alike in the literature (e.g. Fickas and Nagarajan, 1988; Zhang et al., 2005) and the empirical results. For example, the features were analysed by using domain-specific knowledge and experience as well as interviewing experts. However, during the interviews it was noted that there is a difference between analysing the major and minor features. The major features were analysed by using such methods as mentioned in the literature, e.g. feasibility study. Instead, the minor features were not analysed; on the contrary, they were only planned and then considered whether they would fit into the product's content or not. Furthermore, the empirical evidence revealed that analysing features includes following important factors to be noticed: (1) verify understanding, (2) clarify implementing technology, and (3) create cost, time, and work estimations. Analysing features was typically conducted by the experts from different stakeholder groups, e.g. sales and people with technical or strategic perspective.

According to the literature, the product features could be prioritised with formal (e.g. McCarthy, 2003; Phaal et al., 2003b; Groenveld, 2007) and informal (e.g. Phaal et al., 2003a; Blotner, 2004) prioritisation methods. Empirical evidence suggests that features are typically prioritised by using informal prioritisation methods and both functional and non-functional requirements were prioritised at the same time. However, the methods used for informal prioritising differed from the ones proposed in literature. For instance, Blotner (2004) suggests using identity information for initial feature prioritisation in which each feature is presented and input from all team members is collected. According to the interviews, the priorities could be created by using a calculation system; or via a number of customerships per feature or values related to technical importance, market value, or ease with return on investment (ROI), guided prioritisation. Based on the interviews, more information relating to the features was considered during the prioritisation. For example, customer preferences, legislatively or release specific features, and real world matters guided prioritisation. As suggested in the literature (Phaal et al., 2003a), the roadmapping team prioritised the features. However, it was also pointed out in the interviews that the decision could also be made

Table 5
Essential factors influencing product roadmapping.

Phase	Essential factors
Capturing features	Gathering ideas over time and performing interviews to capture features Features come from several sources of information, e.g. market trends and standards Stakeholders include, e.g. collaborators, sales, management, product architects, and market researchers
Analysing features	Using domain-specific knowledge and experience, and interviewing experts Analysing and conducting feasibility study for major features Making a decision whether minor features fit into the product content or not Important factors: (1) verify understanding, (2) clarify implementing technology, and (3) create cost, time, and work estimations Stakeholders include, e.g. a sales representative, and people with technical or strategic perspective
Prioritising features	Using informal prioritisation methods, e.g. a calculation system Prioritising functional and non-functional requirements at the same time Several factors guide prioritisation, e.g. customer preferences, legislative and/or release specific features Stakeholders include, e.g. roadmapping team, product manager, CTO or the person in charge of the product
Roadmap validation and agreement	Validating roadmaps: (1) through negotiations, meetings or reviews to collect input and comments, or (2) through unit's business improvement, customer feedback or product's commercial success Making a roadmap agreement: in a meeting to have a mutual understanding about the product and to make the roadmap official Stakeholders include, e.g. roadmapping team, product owner, product manager, or CTO
Change management of the roadmap	Changes come from delays in product implementation or discovering new or unnecessary product features during product development Using meeting practices to handle change requests and making decisions concerning the change Roadmap's change management process: (1) change request, (2) impact analysis, (3) approve changes and verify understanding, and (4) revise roadmap Stakeholders of a minor change: product manager or CTO Stakeholders of a major change: roadmapping team

by the product manager, CTO or the person in charge of the product. Additionally based on interviews, if consensus could not be reached in the team, the management team of the company or companies made the final decision.

According to both literature (e.g. McCarthy, 2003) and empirical findings, the roadmaps were typically validated and agreed on in meetings. However, according to some of the case companies, the validation occurred when the customers started to buy or not to buy the product. Hence, the validation could also be based on the unit's improvement, customer feedback, or a product's commercial success. The empirical evidence also revealed that the roadmap agreement was made in a meeting together with the roadmapping team to have a mutual understanding about the product and to make the roadmap official. The roadmap agreement could also be approved by the product owner, product manager, or CTO.

Empirical evidence pointed out that the changes to the product roadmaps come from delays in product implementation or when new or unnecessary product features were discovered during product development. It was suggested to use meeting practices to handle change requests and make decisions concerning the change. Moreover, based on the empirical findings, the change management process of the product roadmap consisted of four phases instead of the six phases as proposed in the current literature (Pozgaj et al., 2003). These phases were: (1) change request, (2) analysis of the changes' effects, i.e. impact analysis, (3) approving changes, and (4) creating a new roadmap or revising a roadmap. Additionally, the findings revealed that in the product roadmapping process there was no official CCB during the change management phase as suggested in the literature (Pozgaj et al., 2003). Instead, the decisions concerning the change were made based on the importance of the matter to be changed. Minor changes were made by the product manager or CTO, while major changes were managed by the roadmapping team.

The findings of the empirical research relating to common practices in product roadmapping are summarized in Table 5. The table presents essential factors influencing the different phases of the roadmapping process.

5.3. Critical process phases

The empirical findings indicate that the most important phase of product roadmapping is prioritising features and then the capturing features. The most problematic areas of product roadmapping are prioritising features, managing changes and maintaining roadmaps as well as sharing information, communication, and making a roadmap agreement. The problems in prioritising features were caused by uncertainty about which features should be included in the product or which features should be excluded or left for the forthcoming product releases. To solve this problem, the empirical evidence revealed that prioritisation should be done carefully, with a clear vision in mind and with strategic thinking. Mintzberg (1994) also highlights the importance of strategic thinking by stating that the most successful strategies are visions, not plans. Thus, the difference between planning and strategic thinking should be noticed in order to be productive. The interviewees also suggested some informal prioritisation methods to solve difficulties in prioritising features, e.g. a calculation system. Reaching an agreement was considered difficult, since combining and processing different viewpoints was hard. Meeting practices were suggested to solve this problem. Managing and maintaining roadmaps was though troublesome because of a large number of changing matters. It was suggested that more time should be used in the first phases of product roadmapping (i.e. capturing and analysing features) to reduce changing matters at the end. Also, it was noticed that the more precise the product roadmaps were the harder they were to maintain. Thus, the product roadmaps should only contain the needed information and nothing extra.

6. Conclusions

This article provided an empirical understanding and offered experiences about product roadmapping, which have not been thoroughly explored previously. The main goals of the study were to increase the current empirical evidence on product roadmapping by defining the main stakeholders and their roles during the pro-

cess, organizing the product roadmapping process, establishing the main benefits and challenges faced during the process, and identifying the most critical phases of the process. A research framework for software product roadmapping was created based on the existing literature in order to help to structure and present the research results. The framework provides answers to the research questions and relates to the following elements in product roadmapping: stakeholders, process phases, and the perceived impact of product roadmapping. The research findings increase the product roadmapping knowledge and help companies to develop their own product roadmapping processes. For example, the study identifies the essential factors influencing product roadmapping (Table 5). The findings of the study also help the companies to focus research and development work on the most critical parts of the product roadmapping process, i.e., prioritising and capturing features as well as the change management of the product roadmap. From scientific perspective, the study aims at increasing the body of knowledge in the area of product roadmapping while also identifying future research opportunities in the field of software product roadmapping.

Results of the research indicate that a roadmap is a plan about the company's future directions. It is a leading map where the company is going with its products. Thus, it is the means of structuring and arranging the product development, in order to know how to use certain resources. A roadmap holds the product development together by guiding what is to be done and when. Hence, product development is somehow deterministic and enables steering of the product implementation. Furthermore, product roadmapping is a continuous process, since the roadmapping team has meetings regularly (e.g. biweekly, quarterly, or biannually) in order to create, update or review roadmaps. The product roadmapping team consists of several stakeholders, e.g. at least the following stakeholders are seen as important in the process: product management, marketing, customer and partner representatives, and development. Also, based on the empirical findings, feature management is seen as the key aspect in product roadmapping and accordingly the product roadmapping process is proposed to consist of the following phases: capturing features, analysing features, prioritising features, roadmap validation and agreement, and change management of the roadmap. The most critical phase of product roadmapping is prioritising features. The most problematic areas of product roadmapping are prioritising features, managing changes and maintaining roadmaps as well as sharing information, communication, and making a roadmap agreement.

The research results apply to companies that are involved in software product or service development in the field of ICT industry. In this research, it has not been analysed how, for instance, the products and procedures of the case companies differ from each other and if these have effect on the interviewees opinions or cause some external validity threats for the study. However, all the companies of the study were selected from ICT sector to provide diverse views from the same field. For example, the analysis criteria for analysing features of different types of software products can vary as well as the methods for prioritising features. Also, it should be noticed that the research framework for software product roadmapping is created based on the existing literature at the time of research. Thus, the framework could have been different if it would have been created now, for instance. The limitations of the study also relate to the fact that most of the case companies were large companies with more than 250 employees, thus the results may be applied only to an appropriate extent to smaller companies. Hence, limitations of the study suggest directions for future research, for example the product roadmapping research could be extended to small, and medium sized companies. Additionally, it should be noticed that both the interviewees and the questionnaire respondents have subjective perspectives to the issues asked, thus the answers reveal only the replier's opinions and not the whole opinion of the company.

The product roadmapping research is going to be continued by combining the product roadmapping and agile development methods. That is because a new type of agility is seen important in order to survive in the turbulent and competitive software business environment. Furthermore, as in this research, we have identified that prioritising features and capturing features are the most critical phases of product roadmapping they could be investigated in a more detailed way in the light of agility as well as global software development in the future. Also, through the questionnaire study, topics for the future research were found, for example why and what are the reasons that the change management of the roadmap was considered the least important phase in product roadmapping by the respondents. In this research, this was out of the scope because the focus of the research was set differently, i.e. interviews focused on the most important and problematic areas of the product roadmapping. Also, the aim is to research factors relating to the product roadmapping process itself in order to find out if the process of creating and updating product roadmaps is as straightforward as is typically suggested. For example, McCarthy et al. (2006) have proposed new product development to be regarded as a complex adaptive system (CAS) of decisions. According to them, new product development is seen as more complex than most of the models present it. The intention of further research is also to get more in-depth knowledge of product roadmapping in industry by observation, for instance.

Acknowledgements

This article was written within the Merlin (MERLIN ITEA project 2004–2007, 2007) project, which is an ITEA project. Thus, the authors would like to thank the support of ITEA (ITEA, Information Technology for European Advancement, 2010) and Tekes (Tekes, Finnish Funding Agency for Technology and Innovation 2010). Also, the present authors would like to acknowledge the participants of the survey as well as the reviewers for their comments on the early version of the article.

References

Albright, R.E., 2003. A unifying architecture for roadmaps frames a value scorecard. In: Proceedings of the IEEE International Engineering Management Conference , pp. 383–386.

Albright, R.E., 2002. The process: how to use roadmapping for global platform products. PDMA Visions 26, 19–23.

Albright, R.E., Kappel, T.A., 2003. Roadmapping in the corporation. Res. Technol. Manage. 46 (2), 31–40.

Alreck, P.L., Settle, R.B., 1995. The Survey Research Handbook, 2nd ed. Irwin Professional Publishing, The Irwin Series in Marketing, Chicago.

Arnold, R.S., Bohner, S.A., 1993. Impact analysis—towards a framework for comparison. In: Proceedings of the IEEE Conference on Software Maintenance (CSM) , pp. 292–301.

Beeton, D.A., Phaal, R., Probert, D.R., 2008. Exploratory roadmapping for foresight. Int. J. Technol. Intell. Plann. 4 (4), 398–412.

Blotner, J.A., 2004. PIP: a product planning strategy for the whole family or how we became the Brady bunch. In: Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications , pp. 253–259.

Carlshamre, P., Sandakh, K., Lindvall, M., Regnell, B., Natt och Dag, J., 2001. An industrial survey of requirements interdependencies in software release planning. In: Proceedings of the 5th IEEE International Symposium on Requirements Engineering , pp. 84–91.

DeGregorio, G., 2000. Technology management via a set of dynamically linked roadmaps. In: Proceedings of the 2000 IEEE Conference , pp. 184–190.

Ebert, C., Smouts, M., 2003. Tricks and traps of initiating a product line concept in existing products. In: Proceedings of the 25th International Conference on Software Engineering , pp. 520–525.

Fickas, S., Nagarajan, P., 1988. Critiquing software specifications. IEEE Soft. 5 (6), 37–47.

Gorchels, L., 2000. The Product Manager's Handbook: The Complete Product Management Resource, 2nd ed. NTC Business Books, USA.

Greer, D., Ruhe, G., 2004. Software release planning: an evolutionary and iterative approach. Inform. Soft. Technol. 46 (4), 243–253.
Griffin, A., Hauser, J.R., 1993. The voice of the customer. Market. Sci. 12 (1), 1–27.
Groenveld, P., 2007. Roadmapping integrates business and technology. Res. Technol. Manage. 50 (6), 49–58.
Grynberg, A., Goldin, L., 2003. Product management in telecom industry—using requirements management process. In: Proceedings of the IEEE International Conference on Software—Science, Technology & Engineering (SwSTE) , pp. 63–70.
Holmes, C.J., Ferrill, M.B.A., Phaal, R., 2004. Reasons for roadmapping: a study of the Singaporean SME manufacturing sector. In: Proceedings of the IEEE International Engineering Management Conference (IEMC) , pp. 292–296.
ITEA, Information Technology for European Advancement, ITEA homepage. Available: http://www.itea2.org/ (accessed 18.11.2010).
Jantunen, S., Smolander, K., 2006. Challenges of knowledge and collaboration in roadmapping. In: International Workshop on Software Product Management (IWSPM'06- RE'06 Workshop) , pp. 19–26.
Järvinen, P., 2001. On Research Methods. Opinpajan kirja, Tampere, Finland.
Jiang, T.M., Coyner, M., 2000. Software process disturbances. In: Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC) , pp. 167–168.
Kameoka, A., Kuwahara, T., Li, M., 2003. Integrated strategy development: an integrated roadmapping approach. In: Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET) , pp. 370–379.
Kappel, T.A., 2001. Perspectives on roadmaps: how organisations talk about the future. J. Prod. Innovation Manage. 18 (1), 39–50.
Kostoff, R.N., Schaller, R.R., 2001. Science and technology roadmaps. IEEE Trans. Eng. Manage. 48 (2), 132–143.
Lehtola, L., Kauppinen, M., Kujala, S., 2005. Linking the business view to requirements engineering: long-term product planning by roadmapping. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05) , pp. 439–446.
Li, M., Kameoka, A., 2003. Creating added value from roadmapping process: a knowledge-creating perspective. In: Proceedings of the International Engineering Management Conference (IEMC) , pp. 387–392.
McCarthy, I.P., Tsinopoulos, C., Allen, P., Rose-Anderssen, C., 2006. New product development as a complex adaptive system of decisions. J. Prod. Innovation Manage. 23 (5), 437–456.
McCarthy, R.C., 2003. Linking technological change to business needs. Res. Technol. Manage. 46 (2), 47–52.
MERLIN ITEA project 2004–2007, MERLIN (Embedded Systems Engineering in Collaboration) Project Homepage. Available: http://virtual.vtt.fi/virtual/proj1/projects/merlin/ (accessed 18.11.2010).
Miles, M.B., Huberman, A.M., 1994. An Expanded Sourcebook: Qualitative Data Analysis, 2nd ed. Sage Publications, Thousand Oaks, CA.
Mintzberg, H., 1994. The fall and rise of strategic planning. Harv. Bus. Rev. 72 (January–February), 107–114.
Nardi, P.M., 2003. Doing Survey Research: A Guide to Quantitative Methods. Pearson Education, Inc., Boston.
Nuseibeh, B., Easterbrook, S., 2000. Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, pp. 35–46.
Oppenheim, A.N., 1992. Questionnaire Design Interviewing and Attitude Measurement. Printer Publishers Ltd., London.
Parviainen, P., Hulkko, H., Kääriäinen, J., Takalo, J., Tihinen, M., 2003. Requirements Engineering: Inventory of Technologies. VTT Publications 508, Espoo, Finland.
Phaal, R., Farrukh, C., Probert, D., 2000a. Technology roadmapping: linking technology resources to business objectives. In: Proceedings of the 4th International Conference on Management Innovative Manufacturing (MIMZOOO).
Phaal, R., Farrukh, C., Mills, J., Probert, D., 2003a. Customizing the technology roadmapping approach. In: Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET) , pp. 361–369.
Phaal, R., Farrukh, C., Mitchell, R., Probert, D., 2003b. Starting-up roadmapping fast. Res. Technol. Manage. 46 (2), 52–59.
Phaal, R., Farrukh, C., Probert, D., 2005. Developing a technology roadmapping system. Technol. Manage.: Unify. Discipline Melt. Bound., 99–111.
Phaal, R., Farrukh, C., Probert, D., 2004. Technology roadmapping—a planning framework for evolution and revolution. Technol. Forecast. Soc. Change 71 (1/2), 5–26.
Phaal, R., Farrukh, C., Probert, D., 2000b. Fast-start technology roadmapping. In: Proceedings of the 9th International Conference on Management of Technology (IAMOT) , pp. 1–12.
Pozgaj, Z., Sertic, H., Boban, M., 2003. Effective requirement specification as a precondition for successful software development project. In: Proceedings of the 25th International Conference on Information Technology Interfaces (ITI) , pp. 669–674.
Rautiainen, K., Lassenius, C., Vähäniitty, J., Pyhäjärvi, M., Vanhanen, J., 2002. A tentative framework for managing software product development in small companies. In: Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35) , pp. 3409–3417.
Rautiainen, K., Vuornos, L., Lassenius, C., 2003. An experience in combining flexibility and control in a small company's software product development process. In: Proceedings of the International Symposium on Empirical Software Engineering (ISESE) , pp. 28–37.
Regnell, B., Hösta, M., Natt och Dag, J., Beremark, P, Hjelm, T., 2001. An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. Require. Eng. 6 (1), 51–62.
Richey, J.M., Grinnell, M., 2004. Evolution of roadmapping at motorola. Res. Technol. Manage. 47 (2), 37–41.
Saaty, T.L., 1986. Axiomatic foundation of the analytic hierarchy process. Manage. Sci. 32 (7), 841–855.
Soffer, P., Goldin, L., Kuflik, T., 2005. A unified RE approach for software product evolution: challenges and research agenda. In: Proceedings of the Situational Requirements Engineering Processes (SREP)—Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes , pp. 200–210.
Strauss, J.D., Radnor, M., Peterson, J.W., 1998. Plotting and navigating a non-linear roadmap: knowledge-based roadmapping for emerging and dynamic environments. In: Proceedings of the East Asian Conference on Knowledge Creation Management , pp. 1–26.
Tabrizi, B., Walleigh, R., 1997. Defining next-generation products: an inside look. Harv. Bus. Rev. 75 (6), 116–124.
Tekes, Finnish Funding Agency for Technology and Innovation, Tekes homepage. Available: http://www.tekes.fi/eng/ (accessed 18.11.2010).
Vähäniitty, J., Lassenius, C., Rautiainen, K., Pekkanen, P., 2009. Long-term planning of development efforts by roadmapping—a model and experiences from small software companies. In: 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'09) , p. 300.
Vähäniitty, J., Lassenius, C., Rautiainen, K., 2002. An approach to product roadmapping in small software product businesses. In: Proceedings of the 7th European Conference on Software Quality (ESCQ)—Quality Connection.
van de Weerd, I., Bekkers, W., Brinkkemper, S., 2010. Developing a maturity matrix for software product management. In: Proceedings of the 1st International Conference on Software Business (ICSOB 2010) , pp. 76–89.
van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L., 2006. Towards a reference framework for software product management. In: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06) , pp. 319–322.
VTT, Technical Research Centre of Finland, VTT homepage. Available: http://www.vtt.fi/?lang=en (accessed 1.11.2010).
Wells, R., Phaal, R., Farrukh, C., Probert, D., 2004. Technology roadmapping for a service organization. Res. Technol. Manage. 47 (2), 46–51.
Wiegers, K.E., 2003. Software Requirements. Practical Techniques for Gathering and Managing Requirements throughout the Product Development Cycle, 2nd ed. Microsoft Press, Redmond, WA.
Yin, R.K., 1994. Applied Social Research Methods Series Vol. 5; Case Study Research: Design and Methods, 2nd ed. Sage Publications Inc., Thousand Oaks, CA.
Yoon, B., Phaal, R., Probert, D., 2008. Morphology analysis for technology roadmapping: application of text mining. R&D Manage. 38 (1), 51–68.
Zhang, W., Mei, H., Zhao, H., 2005. A feature-oriented approach to modeling requirements dependencies. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering (ICRE) , pp. 273–282.

Tanja Suomalainen has received her M.Sc. (2006) degree from University of Oulu, Department of Information Processing Science. She received Laudatur from her Master's Thesis about Product Roadmapping in Collaboration. She began her work at VTT Technical Research Center of Finland in 2005 where she worked first as a research trainee and then after graduation as a research scientist. Her research interests include product roadmapping, requirement management, global software development engineering, and agile software development. Currently she is a PhD student in the Faculty of Information Processing Science at University of Oulu.

Outi Salo has received her M.Sc. (2000) and Ph.D (2007) degrees from University of Oulu, Department of Information Processing Science. She first started her career in IT industry as a software analyst (1998–2000). She began her research work at VTT Technical Research Centre of Finland in 2001 where she worked as a research scientist and as a senior research scientist (2001–2008). Her focal research interests focus in software processes, software process improvement, metrics and agile software engineering. After finalizing her Ph.D, she decided to continue building her expertise in industry sector and, since 2008, has been working at Nokia Corporation. She is currently also carrying out studies in order to become a qualified vocational teacher in software engineering.

Pekka Abrahamsson is a full professor of computer science at Free University of Bozen-Bolzano in Italy. He has held academic positions in University of Oulu, VTT Technical Research Centre of Finland and most recently in University of Helsinki. His research interests are centered on empirical software engineering, agile/lean development and cloud computing. He is the recipient of the Nokia Foundation Award in 2007. He currently serves in the advisory board of IEEE Software. He is a member of the IEEE, ACM and ISERN. His practical experience involves five years in the software industry as a software developer and a quality manager.

Jouni Similä is full professor of Software Engineering at the Department of Information Processing Science, University of Oulu and recently until 2010 a Visiting Scholar at International Computer Science Institute and Center for Open Innovation, UC Berkeley. His present research interests include software process assessment and improvement, empirical software engineering, product roadmapping, agile software development, requirements engineering, global software engineering and open innovation. He has authored more than 80 research publications in the Information Systems and Software Engineering fields. Prior to assuming an academic position, he worked in software industry close to 20 years.

PAPER III

Continuous Planning: An Important aspect of Agile and Lean Development

International Journal of Agile Systems and Management,
Vol. 8, No. 2, pp. 132–162.

Continuous planning: an important aspect of agile and lean development

Tanja Suomalainen*, Raija Kuusela and Maarit Tihinen

VTT Technical Research Centre of Finland Ltd,
Kaitoväylä 1, P.O. Box 1100, FI-90571 Oulu, Finland
Email: Tanja.Suomalainen@vtt.fi
Email: Raija.Kuusela@vtt.fi
Email: Maarit.Tihinen@vtt.fi
*Corresponding author

Abstract: Continuous planning is a relatively new and not yet fully studied field of research, especially from the perspective of agile and lean development organisations. To augment the knowledge in this field, this article presents both a literature review and empirical findings from three case studies that reveal how companies conduct continuous planning. The results indicate that continuous planning is not commonly adopted and applied throughout these organisations and that it currently involves only a certain kind of planning (e.g., release planning). The results of this study bring to light that the main elements of continuous planning (i.e., organisational, strategic and business planning) are tightly related to each other and thus should be considered when companies seek to improve their planning processes and practices. The importance of continuous planning will only increase dramatically in turbulent business environments that include ever shorter planning cycles and the need to improve transparency and knowledge-sharing in organisations.

Keywords: continuous planning; strategic planning; business planning; roadmapping; agile; lean; leagile; software development.

Reference to this paper should be made as follows: Suomalainen, T., Kuusela, R. and Tihinen, M. (2015) 'Continuous planning: an important aspect of agile and lean development', *Int. J. Agile Systems and Management*, Vol. 8, No. 2, pp.132–162.

Biographical notes: Tanja Suomalainen received her MSc (2006) degree from the University of Oulu, Department of Information Processing Science. She has worked at VTT (Technical Research Centre of Finland Ltd) since 2005. She began her career as a research trainee and upon graduation became a Research Scientist. Currently, she is also a PhD student in the Faculty of Information Processing Science at the University of Oulu, Finland. Her research interests include continuous planning rating, particularly strategic and business planning, product planning, roadmapping and release planning, as well as agile and lean software development methods and practices.

Raija Kuusela has almost 30 years of experience in the ICT sector. She has been working as a Senior Scientist at VTT since 2009. Prior to this, she worked at Nokia for 14 years in several roles, including software development roles, engineering manager, quality manager, scrum master and product owner. Before her career at Nokia, she worked with other software-intensive

companies for 13 years in software development roles. She received her PhD in Industrial Engineering and Management from the University of Oulu, Finland, in 2007.

Maarit Tihinen is a Senior Scientist at VTT. She graduated in 1991 from the Department of Mathematics at the University of Oulu. She worked as a teacher (mainly mathematics and computer sciences) at the University of Applied Sciences before coming to VTT in 2000. She wrote her secondary subject thesis in 2001 and received her PhD in 2014 in Information Processing Science from the University of Oulu, Finland. She has worked in several national and international research and customer projects and written scientific publications both for international software engineering conferences and journals. Her research interests include various topics related to software processes, global software development, measurement and metrics and quality management.

1 Introduction

Since the mid-1990s, agile methods have assisted software-intensive companies to achieve targets such as decreasing lead times and improving the quality of products. It has been claimed that agile development methods increase the ability of software organisations to respond to dynamic market changes (Highsmith, 2002; Kettunen, 2009). Alternatively, business agility is about changing businesses and business processes, as well as sensing environmental changes and responding appropriately (Overby et al., 2005; Van Oosterhout et al., 2005). Yet another business method, lean thinking (e.g., Womack and Jones, 2003), has been introduced in several industries, including software development (Poppendieck and Poppendieck, 2003, 2007) and in public sector organisations (e.g., in healthcare especially) (Radnor and Walley, 2008). Lean development aims at achieving a continuous and smooth flow of production in pursuance of removing waste in processes and increasing customer value (Womack and Jones, 2003; Petersen and Wohlin, 2010). Charette (2003) argues that lean development means creating a change-tolerant organisation that can survive and succeed in times of uncertainty, change and complexity. Both agile development and lean development concepts have been introduced in the software industry to aid companies in turbulent business environments to achieve shorter lead times (Middleton and Sutton, 2005).

Given that the current business environment of information technology (IT) organisations is very unstable and constantly changing, organisations are increasingly adopting agile and lean development practices. Bellomo et al. (2013) state that both industries and governments have been increasingly adopting agile-based incremental software development practices due to their ability to improve speed. Furthermore, lean development has been proposed as away to achieve substantial cost savings and quality improvements (Radnor and Walley, 2008). Both agile and lean practices have been seen to complement each other. Petersen (2010) compared these two practices and found that:

a both practices share the same goals (i.e., they focus on the customer)

b both practices are defined by similar principles (with the principle of 'seeing the whole' being unique to lean)

c both practices have unique as well as shared principles

d lean does not define processes while agile defines various processes such as Scrum.

Naylor et al. (1999) have defined the term 'leagility' to describe when principles from both agile and lean development are combined in a supply chain strategy. We here use the term 'leagile software development' to denote software development processes that include both agile and lean principles and practices (e.g., Abrahamsson et al., 2002; Womack and Jones, 2003; Petersen, 2010).

The adoption of leagile practices has encouraged organisations to change the way they execute product and service development and business development. Organisations are affected both by external changes (e.g., economy, competition and political interference) and internal changes (e.g., management systems, organisational culture and employee morale). External changes are caused by the above-mentioned agile and lean developments, which have led in many organisations to continuous and iterative product development. Internal changes are caused, for example, by managers' willingness to continuously improve transparency (i.e., the visibility of information to support decision-making in an organisation). Such developments within organisations have led to business environments with more streamlined process structures and continuous competency development. The organisations strategy process also has begun to align more toward a value-oriented and continuous-based planning. Therefore, strategic and financial plans are not created on an annual basis anymore. Instead, the organisations strategy changes based on customer and market needs, execution and the identification of new opportunities. In order to eliminate disconnection between important activities in the organisation, Fitzgerald and Stol (2014) emphasise that the continuous integration of software development and its operational deployment, as well as the connection between business strategies and software development, should be continuously assessed and improved upon. Furthermore, Olsson et al. (2013) argue that software development companies need to move beyond the concept of agile development toward a situation in which software functionality is continuously deployed and customer feedback is the main driver of innovation. Continuous deployment (CD) is the term used to refer to this phenomenon. Although the concept of deploying software to customers as soon as new code is developed is not new and is based on leagile principles, CD expands upon leagility by moving from cyclic to continuous value delivery. CD is about developing the ability to deliver the smallest added value to the customer, which requires automating all processes that must be executed to deliver software to customers (Järvinen et al., 2014). New and innovative approaches that support continuous practices throughout organisations are needed, continuous planning being one of them, to remove disconnection between organisation's important activities.

Continuous planning is about developing planning practices continuously, not just once or twice a year (e.g., Hope and Fraser, 2003). Rickards and Ritsert (2012) point out that even though organisations are expected to have continuous planning practices (e.g., quarterly rolling forecasts and budgets),only a minority of enterprises use them. They suggest using continuous planning instruments instead of traditional, static tools in the belief that environmental changes trigger planning instead of the financial year and thus, that plans should be adjusted according to internal and external events (Rickards and Ritsert, 2012). In order to achieve continuous planning, organisations need to be capable of changing their operations and adapting their mind-sets toward continuous planning and transparency throughout entire organisations.

This article focuses on continuous planning, which is a relatively new and poorly-studied field of research. Hence, the literature on continuous planning is currently inadequate. This research topic is relevant both from the perspective of a leagile organisation, in terms of providing information on methods and techniques for planning and from an agile systems design and evolution perspective, in terms of the creation of content and strategies. The main goal of this article is to identify the current methods and practices of continuous planning in three information and communication technology (ICT) companies. The research questions for this research are as follows:

- How is continuous planning being conducted in agile and lean software development context?
- What are the main benefits and challenges of continuous planning?

This article provides evidence from multiple case studies from which data was collected via interviews, a series of meetings and workshops, as well as through the analysis of company-specific internal memos. The case studies were based on the experiences of three ICT companies: Elektrobit (EB), F-Secure and Tieto. Each of the companies is a large, Finnish-based company with more than 1,000 employees. EB and Tieto operate in the domain of IT products and services, whereas F-Secure operates in the domain of data security. Each of the companies has transformed their organisational practices first in terms of an agile method and then with a lean approach.

In examining continuous planning, this article will focus on its main elements: organisational planning, strategic planning and business planning. Each of these elements is vital and tightly related to one another. Organisational planning defines the organisational level and timeframe of a plan, strategic planning forms the overall plan of an organisation and business planning forms the budgeting frame of a plan. This article will also discuss the implications of continuous planning, including the motivation to implement it and the main benefits and challenges of continuous planning. Companies seeking to develop or improve their continuous planning processes and practices should take all of the components of continuous planning into account to understand how adopting continuous planning can yield significant benefits for an organisation.

This article is structured as follows: in Section 2, continuous planning is defined based on the current literature, including an overview of continuous planning, organisational planning in terms of levels of planning and time frames (i.e., roadmapping), strategic planning and business planning. In Section 3, the research design of this article is presented and the backgrounds of the case companies are described. In Section 4, the research evidence of the empirical studies is given in terms of how continuous planning is conducted in each of the case companies. In Section 5, the research results are discussed, followed by a validation of the study and the limitations of its research results. Finally, Section 6 concludes the paper and directions for future research are given.

2 Related work

In this section, we will outline the current knowledge on continuous planning based on the existing literature and research. This literature review was undertaken to define the current state of continuous planning research and to offer a context for the case studies of

this paper. First, a background on continuous planning is given, followed by the main elements of continuous planning, including organisational planning, strategic planning and business planning. Thereafter, roadmapping is described in greater detail. Finally, the main findings of this section are summarised in relation to a definition of continuous planning.

2.1 Overview

Planning can be understood as consisting of two things: actions and forecasts (i.e., expected outcomes).Whereas forecasting can relate to technology or market trends, planning can relate to products, product lines, resources, or an entire company (Van de Weerd et al., 2010). Continuous planning involves implementing planning practices continuously, not just as part of a top-down annual event (e.g., Hope and Fraser, 2003). Planning should be done continuously so that the full scope of development can be presented at any time (Westkamper and von Briel, 2001). Fitzgerald and Stol (2014) define continuous planning as a holistic effort that involves multiple stakeholders from business and software areas. Planning is understood as a dynamic, open-ended process that evolves in response to changes in a business environment and thus involves the tight integration of planning and execution. In terms software development, continuous planning refers to the organisational capacity to conduct planning in rapid parallel cycles (in hours, days, weeks, or months) depending on the level of planning.

Myers (1999) has stated that continuous planning is required in today's organisations and that it will be increasingly important in the future. The continuous operations of organisations have necessitated the ability to produce open-ended plans that develop and evolve in relation to the dynamics of an environment. Furthermore, incremental planning techniques have also been required to respond to changing situations. In response to these requirements, Myers's (1999) developed the continuous planning and execution framework (CPEF), which sought to combine plan-generation and plan-use capabilities to solve complex tasks in unpredictable and dynamic environments. Continuous planning is taken in the CPEF to be driven by the two following notions: first, plans should be understood as dynamic and open-ended which evolve in response to ever-changing environments. Second, users are understood as integral to the overall planning process in terms of providing inputs that will influence the type of plan that is generated, the number of options to consider, failure assessments and plan-repair strategies.

With the adoption of agile and lean development practices, the practice of continuous planning has evolved toward constant planning in small increments and with more people than is typical of traditional software development methods. Shalloway et al. (2009) have presented a continuous planning process related to software releases performed before each iteration and during daily stand-up meetings. Continuous planning at the project level is done in relation to what is known (e.g., looking two to four weeks forward), the plan is for the next iteration and the work that will be for today. Various industrial experiences (e.g., Lehtola et al., 2007, 2009) of companies have shown that they perform open-ended planning with a pre-defined rhythm. However, while planning can be undertaken at regular intervals, the horizon of the future is not fixed. Company planning is often performed looking only one to two releases ahead, with planning for the near future given greater detail than for the remote future, which is only roughly outlined. Open-ended planning is an effective form of market-driven planning that understands

decisions as involving various trade-offs between now and later (Lehtola et al., 2007, 2009).

It has only been recently realised that planning should be examined from a broader, even more continuous perspective. Continuous planning is not only a project- or team-level activity, but involves higher-level planning as well (e.g., strategy level planning). In the software development context, according to Fitzgerald and Stol (2014), the only forms of continuous planning have emerged from agile development approaches and are related to sprint iterations, or at best, software releases. They conclude that continuous planning is not widespread throughout organisations in the context of software development. Recently, Heikkilä et al. (2013) adopted a three-level planning model, including strategic planning, release planning and operational planning for a large-scale agile software development organisation. Strategic planning involves interaction between business and management and development and is performed over the long-term. Release planning refers to the feature content of the next release and to planning aiming to create content efficiently. Operational planning concerns the implementation of features on day-to-day basis. Heikkila et al. focus on release planning, however, without going into detail on strategic or operational planning. Thus, their understanding of continuous planning lacks a broader perspective.

Several factors of continuous planning can be found in the literature. Koenigsaecker (2009) discusses governance as one of the key issues in a lean organisation's planning process. An organisation's planning process should cycle through each level of leadership returning to the first level of the organisation. Bogsnes (2008) considers organisational planning to be about leadership and creating conditions for good performance to take place, which require an environment of trust and transparency. Accordingly, leaders should work to establish clarity, capability and commitment among their employees. Cosner et al. (2007) emphasise the ability of roadmapping (i.e., a process for documenting the evolution of a company's markets and the product and technology development plans to address those future markets) to link different levels of plans (i.e., portfolio management and finance processes). Roadmaps can be used as guides for skill and competency development as well as for human resources in terms of building competencies that are beneficial to a company's future. In summary, the factors of continuous planning found in the literature include governance, leadership, transparency and competency development.

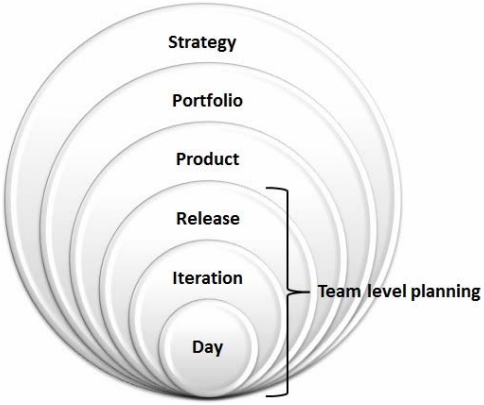
2.2 Organisational planning

The most important aspects of organisational planning are the required planning levels and timeframes (Lehtola et al., 2007). A planning level relates to items that are being planned, whereas a timeframe refers to the time periods of a plan. A company developing and improving its planning practices should define these aspects before initiating aspects of planning. Long-term planning, known as roadmapping, is one solution to bridge the gaps between different levels of planning. According to Cosner et al. (2007) roadmapping should be seen as part of a company's overall business processes and be integrated into its planning cycles. In this section, an overview of organisational planning based on the literature will be discussed in further detail. Roadmapping will be discussed in more detail in Section 2.5.

One of the most important aspects of organisational planning are required planning levels. However, there is no simple answer as to how many levels of planning a company

should have, as both company size and organisational structure play a role in this decision (Lehtola et al., 2007). Agile at the enterprise level requires examining an organisation's entire value stream, management and delivery team planning levels (Shalloway et al., 2009). Cohn (2006) has presented an agile approach to planning called the 'planning onion' that describes the hierarchical relationships between different facets of planning. The planning onion (shown in Figure 1) consists of the following levels of planning: strategy, portfolio, product, release, iteration and day.

Figure 1 Planning onion



Source: Modified from Cohn (2006)

According to Cohn (2006), planning should not extend beyond a planner's horizon. Instead, it should allow time for a planner to pause, examine a changing horizon and make adjustments with a progressively evolving plan. Agile teams achieve this by planning for three distinct horizons: the release, the iteration and daily planning. *Release planning* considers user stories or themes in relation to a new release with the goal of determining the scope, schedule and required resources for a project. A release plan should be updated throughout a project so that it will always reflect the current expectations as to what will be included in the release. During *iteration planning*, which takes place at the start of each product iteration, a product owner identifies the work that a team should address for a new product iteration. *Daily planning* meetings are meant for organising work and synchronising daily efforts. Cohn (2006) explains that product, portfolio and strategic planning commonly exist outside of the concern of most agile teams. While *product planning* involves looking further ahead than the immediate release and planning for the evolution of are leased product or system, *portfolio planning* involves the selection of products that will best implement the vision of a company's strategy. In terms of daily and operational planning, Ruhe (2010) has pointed out that both can become difficult without a proper release plan that is well-aligned with the product and portfolio strategy of a company.

Leffingwell (2011) has presented a framework called *agile enterprise big picture*, which considers both organisational and process models for agile practices. This framework consists of three levels: the team level, the program level and the portfolio level (Leffingwell, 2007, 2011). According to Leffingwell's framework, from an organisational planning point of view, agile development affects, among others, planning and scheduling in software development and in software project management.

The team level planning happens in relation to two aspects of product development: iterations and releases. At the team level, agile team members define, build and test user stories in a series of sprints or iterations (i.e., short time-boxed events) to combine larger, system-wide functionality for release of products, features, or architectural components to external users. Many teams have four to five iterations of development with the cadence of each potentially shippable increment coming roughly every 90 days, which creates a quarterly planning rhythm for an entire company. Furthermore, the responsibility for planning is transferred to the teams. At the program level, planning is accomplished by the release planning function. Release planning is done on a regular basis independent of a project's status and release commitments. At the portfolio level, investment themes relate to the strategic planning horizon, with timeframes spanning from 12 months to over 18 months. Investment themes are used to drive the investment priorities of the company. Therefore, investment themes drive the portfolio vision that is expressed in a series of larger, epic-scale initiatives that will be allocated to various releases over time. Epics express the highest level expression of customer or business needs. Prior to release planning, epics are divided into specific features which in turn are converted in more detailed stories for implementation. Epics are planned in timeframes of 6 to 12 months.

2.3 Strategic planning

A generic strategy process can be divided into four stages: analysis, development, planning and implementation (Eppler and Platts, 2009). Bryson (2011) defines strategic planning as follows: "deliberative, disciplined approach to producing fundamental decisions and actions that shape and guide what an organisation (or other entity) is, what it does, and why". Strategic planning means accounting for where you are, where you want to be, how to get there and how these are connected (Bryson, 2011). It also includes the development of timelines, resource allocations, responsibilities and deliverables (Eppler and Platts, 2009). Many similarities have been recognised between strategic planning and roadmapping. For instance, the definition of roadmapping is quite similar to that of strategic planning (Kappel, 2001) as one element of a strategic planning process (Cosner et al., 2007).

Strategy processes vary across companies, but at the group level, they are commonly continuous and issue-driven (Bogsnes, 2008). The various components of a given strategy commonly consist of specific routines and work patterns that vary from firm to firm and between different types of firms (Nordqvist and Melin, 2010). Te Brömmelstroet (2013) states that strategic planning phases can vary widely in terms of how they are organised, be they bottom-up or top-down. However, all strategic planning processes can be seen as part of a multilevel group process in which planning actors work together toward a shared outcome. Furthermore, Bogsnes (2008) states that during the strategy process, strategic objectives are often defined in what is known as a strategy map. Strategic themes are then commonly addressed as needed, or bi-annual executive committee strategy sessions will be held. When there is a major change in an organisation's strategic direction, its strategic objectives are often renewed or revised.

From a leagile organisation's perspective, Mavengere (2013) discusses strategic agility, which he deems especially important in a competitive business environment. He states that supply chain participants should have their own strategic plans that relate to an entire supply chain's plan, yet he does not go into detail on how such plans are created (i.e., the planning process in detail). Koenigsaecker (2009) details a lean strategic

organisational process in which strategic planning is typically done once per year and is intended as a learning experience. In addition, monthly strategy deployment meetings are held to review progress and create opportunities to share knowledge about lessons learned. The existence of monthly strategy-deployment reviews helps to get a company thinking about how to make its work fundamentally better with each passing month.

2.4 Business planning

Every business should have clearly defined objectives and parameters within which to operate. The business planning process provides an opportunity to assess the range of skills needed for a business to succeed and to identify potential gaps within this range. Financial planning and the preparation of marketing plans help to determine whether or not objectives are being achieved. Wareham and Majka (2003) claim that continuous financial planning processes are commonly based on goals formulated in strategic plans, such as the establishment of capital structures appropriate to an organisation's current competitive and strategic position. They also point out that continuous financial planning should involve a capital allocation process that forces an organisation to prioritise capital spending decisions in a way that will improve services provided while also protecting long-term financial capacity (Wareham and Majka, 2003). Furthermore, marketing plans provide measurable targets to compare and monitor progress, as well as set achievements on a continuous basis (Butler, 2012).

In relation to business planning, Rickards and Ritsert (2012) have discussed rolling forecasts and budgets. The most important characteristics of rolling plans compared to traditional forecasts and budgets are as follows:

- a constant horizon independent of the financial year
- periodicity (the rule of quarterly preparation)
- planning is more detailed in early periods and less detailed in later periods
- planning focuses mainly on monetary and non-monetary business drivers that influence monetary results (revenues and costs).

According to Rickards and Ritsert (2012), rolling forecasts and plans (relating mainly to budgets) are commonly performed over a time period typically spanning five quarters (or 13 months) to eight quarters (or 24 months). Furthermore, they claim that many enterprises that utilise rolling forecast and budgets use them in combination with a traditional budget plan. This approach determines to use the lower boundary of five quarters (or 13 months), because at latest, at the beginning of the fourth quarter (or twelfth month), forecasts and budget values must completely cover the next financial year. Rolling revisions of plans ensure that, the length of the time period covered is constantly evaluated so that new information is integrated into plans. By doing so, more detailed forecasts and budgets can be made for upcoming quarters both near and far in the future.

Hope and Fraser (2003) have presented the 'beyond budgeting management model', in which they define budgeting in a much broader way than is commonly understood. One of the highlights of this model is that it understands budgeting, planning and improvement as a continuous and customised process, not just as parts of an annual event (Hope and Fraser, 2003). According to Bogsnes (2008), similar to the rolling forecasts

and plans presented by Rickards and Ritser (2012), five-quarter rolling forecasts have become a standard in 'beyond budgeting' implementations. However, Bogsnes (2008) clarifies that even though a five-quarter rolling horizon is better than stopping planning at the year-end, planning is still done in a fixed period. Hence, a more dynamic planning process that is more event-based than calendar-driven with no fixed update frequency and with no fixed time horizons should be developed (Bogsnes, 2008).

From an agile organisation perspective, Leffingwell (2011) has pointed out that a set of strategic investment themes drive the organisations vision of all products, systems and services and the responsibility of investment decisions generally lies with a portfolio management team. In most enterprises, investment decisions occur at the business unit-level based on an annual or bi-annual budgeting process. During the budgeting process, the amount of funds available for each business unit or product to invest in development is determined (Leffingwell, 2011). It has been shown (Cosner et al., 2007) that budgeting process should be aligned with the key milestones and events of an organisation's roadmap. More specifically, a budgeting process should include the following three elements; first, research and development (R&D) categories for development resources in each major capability of a roadmap; second, categories of resources for operations that support different capabilities; and third, life-cycle funding plans that forecast necessary 'ramp-ups' as well as simultaneous 'ramp-downs' in operational funding related to different types of capabilities shown on a roadmap.

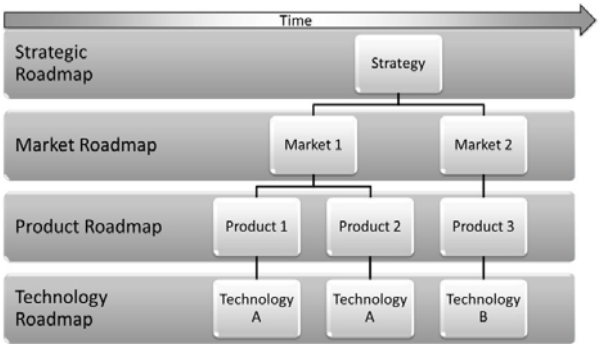
2.5 Roadmapping

In general, a roadmap is a layout of existing routes or paths that is used to decide among different directions toward a desired destination (Kostoff and Schaller, 2001). Roadmaps are forecasts of what is possible or likely to happen and plans that express a course of action (Kappel, 2001). Roadmaps are also intended to be living documents, to be reviewed and updated over time in order to remain useful (Albright, 2003). Roadmapping describes the process of creating and revising roadmaps (Kostoff and Schaller, 2001). Roadmaps can be expressed in various forms, types and with different taxonomies (Kameoka et al., 2003) and should answer a common set of 'why-what-how-when' questions that generally relate to markets, products and technologies (Phaal et al., 2005). A more detailed summary of existing roadmapping processes along with their main areas of focus (i.e., goals and main phases) is presented in Suomalainen et al. (2011).

A roadmap's architecture consists of two dimensions: timeframes and layers and sub-layers (Phaal and Muller, 2009), with each layer of information providing input for the next level (Cosner et al., 2007). Cosner et al. (2007) describe enterprise roadmap development as being divided into the following layers: strategic roadmap, market roadmap, product roadmap, technology roadmaps and enterprise roadmap. Figure 2 describes the interrelationships between these roadmaps. A *strategic roadmap* deals with the long-term objectives of senior management. A *market roadmap* presents known and predicted customer needs along with, for example, competitive strategies, the regulatory environment, complementary product evolution, substitute products and innovations. In a roadmap strategic goals and market targets are defined as milestones or in terms of target dates for certain events. *Product roadmaps* are for documenting performance and feature evolution, as well as presenting new-to-the-company products and new-to-the-world products. *Technology roadmaps* include expected R&D products, their availability dates, driving factors for R&D and related information. An *enterprise roadmap* combines these

different types of roadmaps across an entire enterprise. While each roadmap presents existing plans, the enterprise roadmap may suggest alternate, unfunded plans that could be considered.

Figure 2 Enterprise roadmap development



Source: Modified from Cosner et al. (2007)

Five broad time periods have been recommended for inclusion in roadmaps: the past (and current situation), short-term, medium-term, long-term and vision (Phaal and Muller, 2009). The *past*-timeframe helps to understand key influences and events that have led to the current situation, as well as highlights learning points that will influence the success of future plans. A *short-term* consideration typically looks at a *one-year period* and is the most important output of a roadmap, including tangible plans and committed actions. This timeframe can be understood as a budget horizon for resources to be committed so that actions can be fulfilled. *Medium-term* periods are typically around *three years* and are linked to strategic planning projections that highlight the broader direction and options that influence short-term directions and plans. A *long-term timeframe* is typically a *ten-year timeframe* that represents a bridge between the medium-term strategy and the vision of an organisation. This time frame includes the articulation of key uncertainties and scenarios, as well as the exploration of long-term shifts in technology, business and market environments. In addition, it allows an organisation to capture and assess longer-term issues that may affect current decisions and plans. Lastly, a company's vision relates to knowing where it is going and setting long-term goals.

2.6 Summary

In conclusion, continuous planning involves the continuous implementation of planning practices in rapid, parallel cycles instead of predefined and regular planning occasions. Continuous planning reacts to environmental changes, with internal and external changes triggering planning in addition to a predefined planning rhythm. The timeframe of a plan can vary from hours to months depending on the level of planning. The main elements of continuous planning are as follows: defining the organisational level and timeframes of a plan (i.e., roadmapping), strategic planning to form the overall plan of an organisation and business planning to establish a budgeting frame for a plan.

Examining the literature on the subject, continuous planning is a relatively new and insufficiently studied field of research. The current literature focuses mainly on certain specific levels of planning in an organisation exclusively and lacks a broader perspective

of continuous planning. In relation to what is stated in the current literature, the cases of this study were found not to be applying continuous planning throughout their entire organisations, instead applying it only at certain levels of planning (e.g., release planning, especially in the leagile context). In addition, since strategic and business planning is of less concern to most agile teams, these are not described in detail either in the literature. Literature pertaining to continuous strategic or business planning in from a leagile perspective is particularly lacking at present. In addition, based on the literature review of this study, there is very little empirical research of continuous planning that describes how continuous planning is conducted at different levels of planning. Therefore, this study argues that continuous planning requires greater research with a broader perspective than it has previously been given. The intention of this research is to increase the current empirical evidence on continuous planning and roadmapping both for industries and the field of science to develop improved planning practices.

3 Research design

In this section, the research design of this article is described. The research goals, methods and process will first be introduced, followed by a description of the case companies and data collection methods.

3.1 Research methods and process

The goal of this research was to identify current methods and practices of continuous planning. The research aimed to define what continuous planning means and to determine how industry players perceive it in order to increase the current empirical evidence on continuous planning for players in the fields of industry and science to further research. The research questions of this study are as follows:

- How is continuous planning being conducted in agile and lean software development context?
- What are the main benefits and challenges of continuous planning?

This research includes a multi-case study (Yin, 1994). A case study research approach was decided upon because the literature on the continuous planning is not yet well-formulated and statements on practical experiences in the field of research were difficult to find. Järvinen (2001) emphasises the ability of case studies to examine very complicated circumstances and, in this way, to gather information for the creation of new knowledge. In addition, the case study methodology has been stated to be well-suited for software engineering research, as it involves the study of contemporary phenomena in its natural context (Runeson and Höst, 2009). Furthermore, case studies are especially appropriate when context is expected to play a role in the phenomena of focus (e.g., if the stresses of a given project affect developers' behaviours), or when effects are expected to be wide-ranging or to take a long time (e.g., weeks, months, or years) to appear (Easterbrook et al., 2008), the latter of which was the case in this research.

The research was processed as follows. First, the current state of continuous planning was discerned for each case. Second, empirical research was carried out in which qualitative data was collected through semi-structured interviews (e.g., Järvinen, 2001), a

series of meetings (with case company A) and workshops (in case company C), as well as the analysis of company-specific internal memos in cases A and B (described in more detail in Section 3.2). As described, for example, by Yin (1994), such data collection methods align with the selected case study research method quite well. The interviews were conducted in all the three case companies. The interviews were analysed using the generic process of data analysis presented by Creswell (2003). After each interview, a digital recording was transcribed and thereafter, all data was read through in order to obtain a general sense of the data. A coding process was used to categorise the data and label the categories. The purpose of the coding process was also to generate descriptions for the categories to generate a small number of themes that represented the major findings of the qualitative data. These descriptions and themes were then presented in qualitative narratives in case descriptions (presented in Section 4). The case descriptions were validated by the interviewees by their reading them through and making corrections to them if needed. Finally, based on the empirical research, the current status of continuous planning at each case company was decided upon.

3.2 Case companies

The research included three global ICT companies: EB, Tieto and F-Secure. One of the main reasons for selecting these case companies was because they had transformed their organisational practices to use an agile method (more precisely, the Scrum method (e.g., Abrahamsson et al., 2002) and then complemented this later on with a lean approach (e.g., Middleton et al., 2005). The case companies and the study's respective data collection methods are presented in Table 1.

Table 1 Case companies

Case id	*Company*	*Role of the interviewee*	*Number of employees*	*Industry*	*Start of agile/lean transformation*	*Data collection method*
Case A	EB	Head of quality and environment	1,800	IT, products and services	2007	Series of meetings and an interview with the case company representative; analysing company internal data
Case B	F-Secure	Project manager	1,000	Data security	2005	An interview and analysing company internal data
Case C	Tieto	Scrum team roles	18,000	IT services	2006	Six interviews, nine workshops and three follow-up meetings between workshops

3.2.1 Case A (EB)

Elekrobit (EB) is a global company with roughly 1,800 employees that provides cutting-edge technological solutions to the automotive and wireless industries. The company operates in more than ten countries and has 12 subsidiaries. The company provides development services for wireless products, including mobile phones and

navigation systems, as well as sells radio channel emulators for testing wireless products. EB also offers software for designing and implementing infotainment systems for the automotive market. The EB research data was collected during 2011–2012 via a series of meetings and one interview. In 2011, eight meetings were held to exchange information on continuous planning in which three persons were involved: two researchers from VTT and one representative of the case company. During the meetings, the vital elements of continuous planning were discussed and drafts of the company's continuous planning framework were drawn up. Then, in 2012, a semi-structured interview was held to clarify how continuous planning is conducted at EB, as well as how information is transferred between different levels of the organisation.

3.2.2 Case B (F-Secure)

F-Secure is an anti-virus, computer security and computer software company with approximately 1,000 employees in 20 offices around the world. The company provides software products for digital content protection, such as internet security, antivirus, mobile security and anti-theft software, as well as a range of free online tools as a service through operators. The company has partnerships with more than 200 operators and it operates 22 wholly-owned subsidiaries. The research data on F-Secure case was collected in 2011 during an interview and by analysing the company's internal data. The interview was organised as a semi-structured interview that sought to process the vital themes of the research. The interview lasted one hour and 40 minutes and involved one interviewee and two interviewers. Thereafter, the interview was transcribed and analysed. After the interview, internal memos pertaining to continuous planning at F-Secure were analysed. This data included information related to continuous planning practices that were discussed during the interview, yet this information was nonetheless considered vital in terms of clarifying and verifying the practices and processes of the company related to continuous planning.

3.2.3 Case C (Tieto)

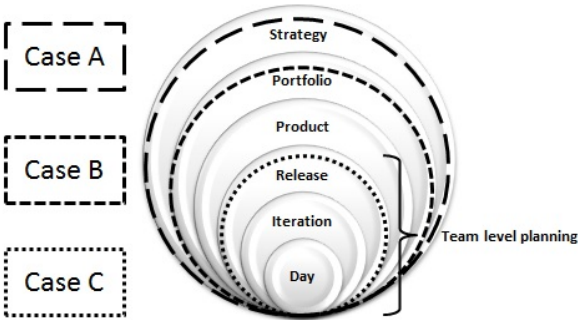
Tieto is an IT service company that provides IT, R&D and consulting services. With approximately 18,000 experts, it is among the leading IT service companies in Northern Europe and is a global leader in certain areas of the field. In this case, the study focused on the sustainability intelligence R&D team (SI team) at Tieto. The research data of the Tieto case was collected mainly in 2010–2012. During 2010–2011, six one-day workshops and three follow-up meetings that used the value-stream mapping (Abdulmalek and Rajgopal, 2007) method were conducted. In the workshops, the team members chose the most critical processes from their perspective, described the current state, as well as identified bottle-necks and improvement actions for their chosen processes. Based on their identified improvements, they then drew a future-state map of the selected processes. In the follow-up meetings, which were conducted between the value-stream mapping workshops, the selected processes and practices were modified by the team when needed. Additionally, six team members were interviewed in the spring of 2011. These interviews focused on exploring what the development methods of the team were and how the team developed the new cloud service. The interviews also revealed underlying needs for continuous planning. Finally, in the beginning of 2012, three

two-hour workshops were organised in which the team members improved the processes to align with the idea of continuous planning.

4 Research results

In this section, the research evidence from the empirical study is presented and the continuous planning processes of each case company are introduced. The research results are presented in terms of Cohn's planning onion framework (presented in Section 2.2.1). By different line styles, Figure 3 illustrates the level of continuous planning predominantly used by each of the case companies.

Figure 3 Continuous planning at the case companies



As can be seen in Figure 3, the continuous planning process in case A begins at the strategic layer. Case B focuses on project and team-level planning in beginning at the portfolio layer and case C focuses on team-level planning beginning at the release layer. Case C's team, however, only verbally communicates at the portfolio and strategy level. The team layer represents the core of the planning onion, with agile teams performing planning at the release, iteration and day layers. The product, portfolio and strategy layers are commonly of lesser concern, because they commonly exist outside the concern of most agile team according to Cohn (2006).

4.1 Continuous strategic planning

In case A, the chief executive officer (CEO) of the company is responsible for the entire company's strategy process and the heads of the company's business segments are responsible for strategy in their respective fields. There were two facilitators at the case company, one for the automotive segment and one for the wireless segment. The motivation for continuous planning at the company stems from the fact that plans cannot be fixed for one year ahead anymore. Instead, plans must be able to change continuously, as the business environment is constantly changing. Thus, the evolution of planning toward continuously iterated and updated product and service planning and roadmaps was needed. Furthermore, even though a strategy exists at all times within the company, it is now iteratively and continuously updated based on market and customer demands. Past financial crises have also caused the company to realise the importance of continuously planning ahead both from an operational and financial perspective.

The need for continuous planning practises also relates to transparency in order to share information. It had been realised by the interviewee that the transparency and continuous planning of all the activities throughout the organisation were needed. The company needs continuous visibility of its development and operations in order to share and provide information with all employees. On the one hand, operational transparency is related to increasing the visibility of the performed work, plan and executed actions in relation to the fulfilment of one set of defined criteria. On the other hand, development transparency is related to identifying the potential technological and cultural barriers to implementing increased transparency and improving learning in connection to needs that arise due to increased transparency. Also, knowledge sharing had become increasingly important inside the company. The representative for case A defined in one of the meetings that "both push (e.g., trainings) and pull type (e.g., publishing information via wikis and articles) of knowledge sharing is needed". Furthermore, the representative clarified that, beyond this visibility, continuous competency planning and development are needed. Employees' competencies had to be able to adapt constantly and change through continuous analysis, development activities and evaluations as to the successfulness of actions.

The continuous planning process was launched in November 2010 at case A, including financial and strategy planning that adopted many of the 'beyond budgeting' model's principles. At first, the strategic actions were walked through, defined and reviewed on a quarterly basis. Next, the intention of the company was to have a continuous strategy that would be continuously connected to the goals established in bi-weekly reviews with different teams. Based on the interview data, two main *levels of continuous planning* were recognised: financial planning and strategic planning. Each of these levels involves approximately one fourth of the personnel in the business segments in the irrespective planning processes.

The *financial planning* means that the financial framework is conducted only once a year. Thereafter, the planning is continuous. The case company's change toward continuous financial planning began in 2012 and there has not been bi-annual budgeting since then. Continuous financial planning means that there is a continuously available financial forecast. For example, currently the company has a rolling forecast for budgeting until the end of the next year. The framework is created at the end of each year (in November), which includes a budget overview of the rest of the following year. The idea of the financial planning is constant in that actual expenses are continuously considered and compared to budgets. Thus, financial actions involve each of the budget items in the short-term and initiatives for the long-term. For example, planning training, which constitute expenses (e.g., common and travel), are budgeted and forecasted.

Strategic reviews are done quarterly and are visible at different levels of the company (e.g., business, market, technology, processes, roadmaps, operational modes, partnerships and action plans). These different levels also have a financial perspective. Strategic planning and financial planning are closely tied together because they both are visible at different levels of the company.

Currently, *the strategy process* is constantly rolling, which means that the strategy is continuously planned. The main phases of the strategy process are as follows:

1 strategy review

2 strategy development and acceptance

3 strategy communication and execution.

The main actions of the business segment's strategy process are as follows:

1 a strategy health check

2 vision and strategy guidance and direction, which includes strategy and roadmap planning, as well as long-range financial planning for the next three years, plus the current year

3 communication of the strategy and execution of the action of the strategy.

The timeframe for the long-term strategy, roadmap and financial planning is the approaching three years plus the current year. The strategic or financial framework for the next year is established prior to the end of the year in a short planning cycle. Thereafter, the framework along with the plans is reviewed quarterly and monthly. The continuous strategy process, along with the phases and their actions, are shown in Figure 4, which was drawn based on both the data from the interview and the company-specific memos.

Figure 4 Continuous strategic planning (case A) (see online version for colours)

A *strategy health check* is done annually at the end of the year from December to January, during which period of time the current status of the strategy is reviewed. Needs for major strategy updates (e.g., establishing a new business area) are addressed in January. At the end of January, strategic work is considered in relation to initiated strategy elements that need to be updated and plans are made regarding how to proceed with the strategy, especially in terms of changed elements.

The *vision and strategy guidance and directions* are planned for a timeframe of the forthcoming three years plus the current year. The strategy, roadmap planning and financial planning are done from a three-years plus one-year perspective. Thus, the company's strategy is always understood at all times as a whole, yet its various elements can be changed at any time. In the beginning of the year, changes to the strategy are reviewed in a phase known as *the strategy creation and elements update.* New strategy element updates are approved in April (by the segment board), followed by board strategy acceptance. The updated strategy is communicated to the employees in May. The

s*trategy communication* is the only action that is performed as an exact time yearly. Otherwise, planning was considered continuous.

To summarise, the strategy frame is reviewed annually and roadmaps are updated quarterly. Plans are reviewed quarterly and at the same time strategy-related outcomes are reviewed. Management reviews the strategy goals quarterly or when needed. In the quarterly strategy review, the following aspects are checked: the current status, the latest updates to strategy and progression of the strategy deployment. After the strategy review, the review results are re-reviewed by the company's business areas. Then, as the interviewee put it, "it is explained how we have progressed with the goals, where we have lagged behind, and what we have unfinished".

There are two perspectives to the strategy. First, the strategy divides into several roadmaps that are made public and available as online information. Therefore, the strategy is visible to all within the organisation. The business roadmap defines, for example, what new information management services are planned to be released and in which quarter and what their status is. These roadmaps are constantly updated. Second, all the issues and goals related to the strategy are made visible via a tool called Jira™ (a commercial software product). Therefore, outcomes can be reviewed in real-time.

4.2 Continuous portfolio and product planning

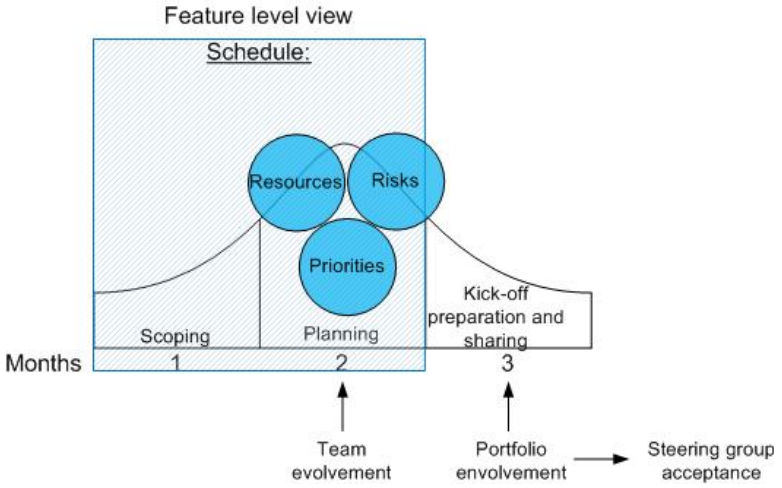
In the case B, the interviewee was involved in a global project operating in three sites. The project was seeking to come up with an overall solution that involved background systems and several clients in close cooperation. The interviewee's role was project manager. The need for continuous planning was recognised in the company given the fact that its business and R&D units did not work together as well as expected and tensions between the primes could be pinpointed. Furthermore, it had been R&D's long-term wish to achieve shorter planning cycles. Therefore, in the beginning of 2010, Dean Leffingwell (author of the agile enterprise big picture, presented in Section 2.1) was invited to coach the company's planning.

Based on the data from the interview and the company's internal memos, the following levels of planning could be defined as existing at case B: *portfolio, project (i.e., features) and team (i.e., stories)*. For smaller projects, only the portfolio and project levels exist. Projects are managed by handling features and teams are managed through feature stories. The team is comprised of a two-levelled backlog: a feature level and a story level. The higher level of the two, the feature level, is defined during planning and involves features being split into stories for implementation in teams. Continuous planning in case B means planning in terms of three-month intervals. However, the company does extend its foresight forwards up to six months in terms of some kind of deliverable and scope.

Continuous planning at the project level is conducted as follows. The most important issue for the team is the work amount, which can range from one to four sprints (i.e., time-boxed effort). Another important issue is customer value, which means that a customer can understand what the team is doing. A project plan is one of the team's outputs, which is co-created so that everyone can see the same artefact and suggest improvements. The project plan includes a scope, resources and schedule. The schedule for the project includes a scope and resources that are each separately planned. The interviewee explained as follows: "So, first, we make estimates, then we make allocations to resources, and finally, we have a plan. That is, all the important parts of the project:

scope, schedule, and resources". The main activity of continuous project planning is balancing these three elements, which also serves to establish the priority of the items in the scope. The main working method of the team is 'war-room workshopping', which enables proper communication and less confusion, improves team spirit and enables visual planning. The main phases of continuous project planning include scoping, planning and kick-off preparation and sharing, all of which are illustrated in Figure 5.

Figure 5 Continuous project planning (case B) (see online version for colours)



The phases of continuous project planning (Figure 5) areas follows:

1 In *scoping*, the features are described and listed and then available teams are identified. The next step involves the estimation of the effort required for specific work items, where the team estimates how many sprints are required for each item.

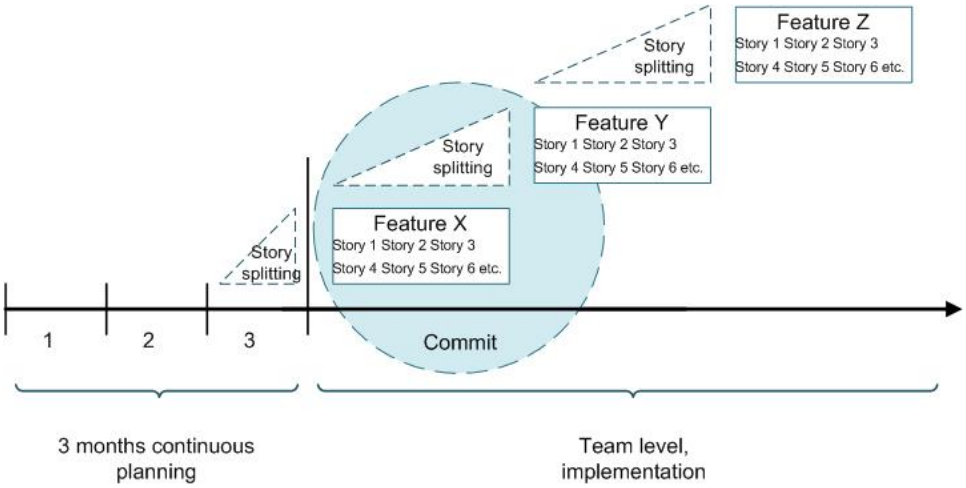
2 In *planning*, the actual feature level is created. The team discusses the features, organises them on the wall of the workshopping war-room, complements their descriptions and then prioritises them. The left side of the wall shows the most important features and the right side the less important features. The priorities are also divided according to required competencies and the project manager then plans the required resources for the project. The resourcing is then negotiated in relation to the portfolio and human resource management and the planning office is responsible for ensuring all of the teams' allocations to projects. Teams are selected based on a given project's business cases. The planning also considers constraints related to resourcing and more explicitly, associated risks and priorities. Risks are identified on separate areas of the war-room walls. Anyone in the team is allowed to post risks as well as to resolve them. Risks are discussed continuously, with the main idea being to continuously change the project plan to eliminate risks. In sum, in order to eliminate risks the team either includes mitigation as part of the scope of a project or changes its approach so that risk probability is reduced. Thus, the interviewee pointed out that *risks feed planning actions and changes to the plan*. The team is openly invited (during a specified timeslot) to see the content and resources of the next project, which calls for team evolvement.

3 In *kick-off preparation and sharing,* the project plan, including features and risks, is presented by a team member to the portfolio owners (i.e., project business directors and vice presidents), which calls for portfolio evolvement. The team then asks for approval or requests changes from the steering group. If changes to project plan are needed, another meeting is held after one or two weeks, or otherwise, the team progresses with the plan. The project portfolio steering group includes all of the required persons, including those at the executive team level. Finally, once the plan is approved by all involved, the plan can be shared with other teams.

4.3 Continuous team-level planning

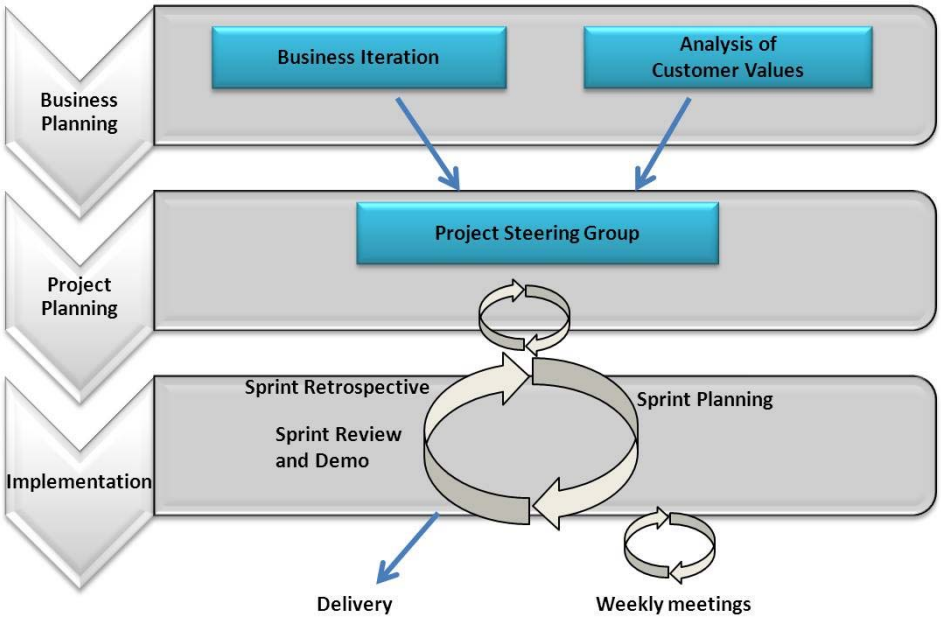
In case B, the team performs continuous feature planning on a weekly basis to keep its priorities and visibility of the team's status up to date. The continuous planning of features is performed in the form of *a feature priority queue*, which involves the creation of a row (or queue) of features in order of priority (with the most important features on the left) and the address of these features, primarily to acquire an understanding of a feature's size. This planning is then completed with resourcing capacity so that each feature is matched in terms of size from left to right, starting with the highest priority. Size is determined by 'team sprints' (i.e., how many full sprints by a single team will be required to complete a feature). This team level planning focuses on the essence of planning prioritising and estimating. When an understanding of this is achieved, planning can be conducted continuously and iteratively. As features are completed on the left, they are removed from the queue. New items are entered in a specific position into the priority queue according to the priority they are assigned. This can be planned and re-planned at any time to any degree. Continuous feature planning does not mean that you need to change everything all the time. If a change is needed, a set of features may be fixed. Simply by monitoring progress and recognising that work remaining in the queue matches with the capacity available constitutes continuous planning. The team level continuous planning of case B is presented in Figure 6.

Figure 6 Continuous team-level planning (case B) (see online version for colours)



In case C, continuous planning was done at the team level. The team used an agile method (Scrum) and later adopted lean principles. After adopting these leagile practices, the team recognised that the time-boxed and static backlog for sprints was not working optimally. When it was realised that urgent tasks were continuously being presented to the team, this forced it to consider root causes for this and what to do in order to control its work. As a consequence, the team headed toward implementing continuous planning; in parallel with their development activities, the team members also contributed to business planning and strategic planning by introducing new and innovative services to the company's portfolio. The team level planning in case C is illustrated in Figure 7.

Figure 7 Continuous team-level planning (case C) (see online version for colours)



The continuous planning approach in case C was comprised of the following main activities: *business iteration, consumer value analysis,* the project's management and planning by *a project steering group and implementation*. Additionally, some dedicated team members actively participated in business iteration and consumer value analysis, which helped the team to keep up-to-date in terms of service development. While, the team were informed of new requirements in terms of businesses' and consumers' values and preferences, they could merge the team's new ideas into business plans. Most team members also participated in the project steering group meetings.

Business iteration activities involve a series of sprints (i.e., it considers a timeframe that is long enough to enable the production of a release plan that is useful to a business). Business iterations aim at fulfilling business goals and involve results being reviewed and adapted to a larger, business-oriented plan. *Consumer value analysis* contains three phases:

1 The survey of consumer values and preferences in a given business context. The intended output of this phase is to create persona dimensions (i.e., personas) and customer profiles. Additionally, descriptions of the main consumer profiles/personas can be used as criteria for tailoring value proposition, fine-tuning marketing messages and improving user experiences.

2 Design implications (i.e., design for end users), which aims at understanding what personas and market segmentation (based on values) mean for a company's design processes, marketing messages and experience creation.

3 Mapping product-related customer value (i.e., design with end users), which aims to understand end users' desired, expected and received value related to a service under development.

A *project steering group* is a governing body of a project that meets once during each quarter of the year. The group reviews and accepts/rejects the products of business iterations and consumer value analysis. Participants in the group include a product owner, a user-experience specialist, a solution manager, a technical manager, a agile/lean mentor, a consumer value analysis team and a Scrum master. The project steering group organises its activities in order of greatest effectiveness for the project. The group performs problem-solving by informing upper levels in a company of possible problems if the group itself cannot solve them itself. Lastly, *implementation work* is conducted in two-week sprints following general Scrum practices, such as sprint planning, daily meetings, sprint reviews and demos, sprint retrospectives and continuous integration.

4.4 Benefits and challenges of continuous planning

With regard to case A, from a financial perspective, the main benefits of continuous planning were that financial estimates will become more accurate. From a strategic perspective, the interviewee highlighted "we are able to show to the staff current stages of the status, for example, what we have done and where we want to be now, at the mid-term, and in the long-term". Since case A's strategy became reviewed, followed-up, communicated and published quarterly, it has led to increased trust in management, as well as to improved competitiveness and decreased fixed costs. Alternatively, in case B, in terms of R&D project management, a key benefit of continuous planning has been that it has decreased work-in-progress as well as improved quality levels. The team has learned how to write features because project performance times are now much clearer than they were before. Feature level planning has also become more reliable given the features have not been much modified during the implementation of the product. Second, from a product management perspective, continuous planning has enabled a better level of communication and brought many more people into the planning process. Third, from a portfolio level perspective, a key benefit has been that the people in charge at the portfolio level are now able to come to the war-room and actually see what their team is currently doing because the team progress is presented in the war-rooms wall. This visual wall also makes it much easier to explain why e.g., features and sprints have been done a certain way instead of trying to explain them using a backlog on an Excel spreadsheet in which context, including risks and resources and other additional information, is lost. Furthermore, according to the interviewee, it is quite easy to put additional information on the war-rooms visual wall.

In case A, the main challenges of continuous planning relate to people (e.g., their willingness to be engaged and involved) and to transparency (e.g., how to make goals visible and place them in a continuous setting). Some people may view continuous planning as leading to a loss of authority among employees. However, it is less about authority or control and more about transparency of information. The general attitude toward continuous planning needs to be embraced throughout an organisation. The main expectations of continuous planning are related to transforming goals into a mode of continuous planning. In case B, the interviewee named several challenges related to continuous planning. For example, one challenge was how to fulfil long-term issues. When long-term goals are taken into account by the management, they should be quickly optimised into short-term business productivity by the team. Due to this, the interviewee posed the following question: "are we taking the longer term architecture and business strategies enough into account?" It was pointed out that continuous planning does not remove the need for long-term planning when the interviewee stated the following: "We need all the time to think of our resourcing needs and our competency needs and location strategy, technology strategy, and all those are of course going on in the background". The portfolio level, for example, was not clearly visible to the team. Only once every three months does someone specify what kinds of projects are about to begin and there were not any tools being used for presenting portfolio-level plans. One of the main obstacles of continuous planning was that many parts of the firm would rather work with long-term plans than three-month plans. For example, the business areas in the case company B wanted a view from six months to one year, but in order to conduct continuous planning, the team needed to state that they could only promise a three-month view, with the long-term view representing only a rough outlook. It was noted that a short-term planning cycle requires a significant mental shift for many actors, especially for those outside the field of R&D. It was pointed out that a team needs to be aware of incremental development practices in general and their capability to do work must be visible at all times to the rest of the company.

5 Discussion

In this section, the research results are discussed and compared with the existing literature. First, the empirical findings from the organisational, strategic and business planning are presented. Second, roadmapping from the perspectives of the case companies will be discussed. Finally, the implications of continuous planning are presented along with the validations and limitations of the research.

5.1 Organisational, strategic and business planning

Continuous planning is comprised of organisational, strategic and business planning. As pointed out in literature (Lehtola et al., 2007), the most important aspects of organisational planning are established levels of planning and timeframes and the number and different levels of planning within companies will vary according to size and organisational structure. Evidence of these points was found out in this research. In comparing the three case companies of this study, it can be concluded that they each are relatively different: continuous planning was considered at different organisational levels in each of the cases, which played a role in the differing timeframes of each case's plan.

Continuous planning was not applied throughout any entire organisation, but was rather focused at a specific level of planning. In case A, two main levels of continuous planning were recognised: strategic planning and financial planning. In case B, three levels of organisational planning were defined: portfolio, project (i.e., features) and team (i.e., stories). However, smaller projects focused only at the portfolio and project levels. The continuous planning process was mainly described as existing at the project and team levels. In case C, continuous planning was mainly at the team level. However, team members contributed to business and strategic planning when introducing new and innovative services to the company's portfolio. In both cases B and C, continuous planning was conducted by team members, whereas in case A it was more of a management effort.

Both the literature and the empirical findings of this study find strategic and financial planning to be closely related. According to the literature, a continuous financial planning process is based on goals formulated in a strategic plan (Wareham and Majka, 2003). According to the case studies of this paper, strategic planning is visible throughout an entire organisation, as each of the different levels of plans (e.g., business, market and technology) have a financial perspective on the plans. In case A, long-term strategic and financial planning (as well as roadmapping) were done for a projected three-year timeframe that included the current year. The strategic and financial frame for the following year was developed prior to the turn of the year and thereafter was reviewed on a monthly and quarterly basis. Wareham and Majka (2003) have stated that companies understand financial planning as a capital allocation process in which actual expenses are considered and compared to budgets, where actions are involved in each budget item in the short-term and as initiatives in the long-term.

In order to achieve continuous planning, the case company A adapted many of the 'beyond budgeting' (Hope and Fraser, 2003) principles. They realised that budgeting, planning and improvement had to be recognised as a continuous process, not just an annual event. Financial planning had to be seen in broader context, as suggested by Hope and Fraser (2003), as part of a performance management process. In fact, in this case, financial planning also had to involve continuous roll-over periods rather than biannual assessments. The financial framework of this case was assessed only once a year, yet thereafter its planning was continuous. The continuous financial planning process in case A became similar to the budgeting and financial planning process presented by Rickards and Ritsert (2012).

In case A, the company also changed its strategic actions toward continuous planning in terms of planning and reviewing strategies on monthly or quarterly basis. According to Koenigsaecker's (2009), planning should be done once per year and strategy deployment meetings should be held monthly thereafter. The strategy process at the case company had three phases: review, development and acceptance and communication and implementation. Respectively, four phases of strategic planning were mentioned in the literature (Eppler and Platts, 2009): analysis, development, planning and implementation. The communication and continuity of the planning process (e.g., how often the plans are reviewed and revised) were not highlighted in the literature as they appeared in the case. The literature discusses how strategic plans should be adaptable to internal and external changes (Nordqvist and Melin, 2010) and that when there is a major change in strategic direction, strategic objectives should be renewed or revised (Bogsnes, 2008). As shown in the case studies, benefits (e.g., more accurate estimates, improved competencies, decreases work-in-progress) can be achieved at all levels of planning. Therefore,

organisations should identify their own continuous planning practices (in relation to the planning onion framework) and at what levels to implement them in order to yield greater benefits.

In conclusion, as a somewhat new and insufficiently studied field, the literature on continuous planning is inadequate as it exists today, especially from a leagile development perspective. The current literature focuses mainly on a specific or single level of planning in an organisation while overlooking a wider perspective of continuous planning. In reference to the current literature, this study found that continuous planning is not being implemented throughout the entirety of organisations but rather applied only at a certain level of planning (e.g., release planning). In addition, product, portfolio, strategic and business planning were found to be of little concern to most agile teams. This study thus highlights that continuous planning should be examined from a broader perspective.

5.2 Roadmapping

The case studies reveal different timeframes for plans, with each timeframe directly related to the level of a plan in question. Continuous planning could be linked to long-term planning in only one of the cases, namely, case A, in which long-term plans projected three years ahead. Otherwise, the plans of the cases were reviewed on a quarterly or monthly basis at the strategy level and reviewed quarterly at the project level. However, team level plans were reviewed biweekly or weekly. Continuous planning was predominantly considered as short-term planning, though long-term planning (e.g., strategic, business, market and portfolio plans) was consistently considered at the higher levels of the enterprises.

In case A, the timeframe for long-term strategy, roadmap and financial planning projected three years forward, including the current year. The strategic or financial frame for the following year was established prior to the turn of the year and thereafter, along with further plans and roadmaps, was reviewed on a quarterly and monthly basis. Similar to Rickards and Ritsert (2012) discussion, planning was found to be more detailed in early periods at the beginning of the year and less detailed in later periods of the year. The continuous planning in case A also related to the strategic roadmap layer presented by Cosner et al. (2007). Furthermore, as pointed out by Phaal and Muller (2009), the strategic planning horizon was found to consist of mid-term planning (roughly three years into the future), which highlights the broader direction and options that influence the short-term directions and plans. The short-term plans are in this case the quarterly and monthly reviews. In the case A, the planning strategy was divided into several roadmaps and as a business roadmap been one of them it defined company's current services and their status. Both the strategies and roadmaps of the organisations were made visible and available as public information via a tool called Jira, which allowed them to be reviewed in real-time.

In cases B and C, the timeframe for continuous planning at the project level was three months. In case B, there was also visibility for the forthcoming six months, including deliverable(s) and scope. In case B, continuous planning did not remove the need for long-term planning, as resourcing and competency needs, as well as location and technology strategies, had to be considered at all times. Continuous planning at the project level in cases B and C related to the product roadmap layer presented by Cosner et al. (2007), as it mainly involved documenting performance and feature evolution, as

well as presenting new products to the company and the world. According to Phaal and Muller (2009), short-term roadmaps are commonly set using one-year projections, yet in this case the timeframe was even shorter at three months. As highlighted by Phaal and Muller (2009), short-term roadmap is the most important layer because it includes tangible plans and committed actions. Furthermore, short-term roadmaps can be called the budget horizon, as resources need to be committed so that actions can be fulfilled. Case B also showed that the main tasks of project level planning involved making estimates in terms of scope and schedule and afterwards allocating resources. At the team level, the team of case B employed an even tighter planning cycle of continuous feature planning than case C, which was done on a weekly basis to keep priorities and visibility up-to-date. Such priority planning in terms of features can be linked to the technology roadmap layer presented by Cosner et al. (2007), as it included expected R&D products, their availability dates, motivating factors for R&D and related information. Similarly, in case C, the timeframe of continuous feature planning and development was set at the team level and involved two-week sprints, relating it to the same layer of enterprise roadmap development as case B.

5.3 Implications of continuous planning

According to both the literature (e.g., Hope and Fraser, 2003; Rickards and Ritsert, 2012) and the empirical evidence of this study, the motivation for continuous planning arises from the fact that organisational plans increasingly cannot hold a fixed-focus of one year ahead anymore. Instead, plans need to be revised continuously according to consistent changes throughout the financial year. According to Rickards and Ritsert (2012), environmental changes trigger enterprise planning, not the financial year and thus planning frameworks need to be constantly adjusted to internal and external events. The interviews of case A speak to the fact that business environments are in constant flux. Thus, enterprises must adapt to change and look to benefit from the opportunities such changes may offer. Financial crises of the past decade have also caused companies to rethink approaches to planning and lead to their realisation of the importance of continuously planning both from an operational and financial perspective. Furthermore, an enterprise's internal problems (e.g., tensions between business and R&D, as well as developers' long-term goals to achieve shorter planning cycles) were found to drive the need for continuous planning. In the workshops of case C, the development team realised that a time-boxed and static backlog for sprints was not the best way of working, as certain priority tasks would emerge during the sprints of which a standard model of working could not accommodate. The development team was interrupted with urgent tasks that forced it to consider new ways of managing its work. The team realised that its plans needed to evolve into a continuous planning approach.

Based on the meetings and the interview of case A, motivations to implement continuous planning were also found to stem from the need to improve transparency and knowledge sharing. Continuous visibility of development and operations was found to be needed in order to share and provide information more effectively. Operational transparency was found to increase the visibility of the performed work, plans and executed actions and development transparency was found to identify potential barriers to both the technological and cultural implementation of transparency and improving learning. With increased visibility, the need for continuous competency planning and development became apparent. It was found that competencies had to be able to adapt

constantly and change via continuous analysis, development activities and evaluating the successfulness of actions.

The results of the empirical study confirm the findings of the literature (Cosner et al., 2007; Bogsnes, 2008; Koenigsaecker, 2009) that there are several components of continuous planning, including governance, leadership, transparency and competency development. Based on the empirical evidence of this study, the factors of continuous planning were found to include transparency and knowledge sharing, competency development and human aspects. While mentioned in the literature, governance and leadership issues were not found in the case studies, although it was pointed out in one of the cases that continuous planning had improved trust toward management.

5.4 Validations and limitations of the research

The case companies of this study are companies involved in IT product and service development that have transformed themselves into agile and lean enterprises. Each of the case companies is a large company with more than 1,000 employees. In order to establish the reliability of this study, triangulation of the study's qualitative data was performed as a validity procedure. The purpose of triangulation is to approach a studied object from various angles to provide an in-depth understanding of it (Runeson and Höst, 2009). The research results are based on three case studies in which data was gathered through interviews, a series of meetings and workshops and through the analysis of company-specific internal memos. The data (i.e., source) triangulation of this study served to strengthen its research results, as more than one data source was used and the same kind of data was collected on different occasions. The results of this study were also strengthened by observer triangulation. There were altogether two researchers involved in this research and the representatives of the case companies possessed various different roles. The research results are also bolstered via the triangulation of theory in his study, as alternative theories were discussed in the presentation of the research results.

The limitations of the study are as follows. The results of this study can only be applied to a certain extent to smaller companies, as all the study's case companies are considered to be large. Furthermore, the reliability of the results is weakened due to the fact that there was only one interviewee per company. The interviewees also possess subjective perspectives as to the questions asked concerning how the continuous planning process is viewed and defined within each case company. Lastly, the interviewees' answers do not speak to the collective opinions of others in the case companies.

6 Conclusions

This article has sought to draw attention to continuous planning, which is a relatively new and poorly studied field of research. During this paper's literature review, it was determined that there are only a few articles on continuous planning that can currently be found and few of these are written from an agile and lean development perspective. Thus, the main goal of the study was to increase continuous planning knowledge in terms of both the literature and empirical evidence. The empirical evidence of this study was

formed drawn from the experiences of three case companies in terms of how they viewed and conducted continuous planning.

The research findings highlight the importance of continuous planning throughout an entire organisation including the elements of continuous planning (organisational planning, strategic planning and business planning) and their tight interrelation. Organisational planning serves to define a plan's organisational level and the timeframes of a plan, strategic planning serves to set an overall plan of an organisation and business planning serves to establish the budgeting frame of a plan. Furthermore, it has been here discussed that continuous planning may not be possible throughout an entire organisation and may only engage a certain level of planning. While continuous planning in agile and lean organisations commonly relates to release planning, this research has shed light on a broader perspective than this by defining continuous strategic and financial planning as well as continuous project and team-level planning. Companies seeking to develop or improve their continuous planning processes and practices should take each of the elements of continuous planning into account from a broader perspective to better understand the benefits it can yield for an organisation. The results of this research also reveal that companies should consider other aspects related to continuous planning, such as leadership, transparency and competency development in order to enable and succeed in their continuous planning efforts.

The motivation toward continuous planning has arisen from both external and internal challenges that companies face in today's volatile market environments. There is a clear need for continuous planning, as organisations face difficulties in developing long-term plans due to constant changes in their customer and market-bases, as well as in product and technology development. Moreover, recent financial crises have caused companies to rethink their approaches to planning and to realise the importance of continuous planning both from an operational and financial perspective. Continuous planning involves creating and revising plans as needed, typically more often than once a year. Based on the empirical findings of this study, the majority of long-term plans looked three years ahead. At the strategy, business, or project levels, plans were reviewed quarterly, whereas at the team level plans were reviewed biweekly or weekly. While continuous planning was mainly understood as short-term planning, this did not remove the need for long-term planning, as strategic, business, market and portfolio planning had to be constantly considered at the higher levels of each enterprise.

Future research on continuous planning should seek to collect more and broader case data from several different companies at different organisational levels. The cases of this study defined continuous planning at the strategy and financial levels, as well as project and team levels, yet continuous planning at business and portfolio levels was not discussed at length. Hence, continuous planning research should further examine these levels. Furthermore, future research could examine how continuous planning is conducted at each layer of each organisation in this research (e.g., how is continuous planning conducted at the product and team levels at the company of case A, which in this research focused mainly on the strategic and financial planning levels. Lastly, research methods other than interviews and workshops (e.g., observation) should be used to gather more in-depth knowledge of continuous planning in industries. The existence of broader case data would allow for comparing and contrasting the attributes of different cases.

References

Abdulmalek, F.A. and Rajgopal, J. (2007) 'Analyzing the benefits of lean manufacturing and value stream mapping via simulation: a process sector case study', *Int J Prod Econ*, Vol. 107, No. 1, pp.223–236.

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002) *Agile Software Development Methods: Review and Analysis*, VTT Publications 478, VTT Technical report, Espoo.

Albright, R.E. (2003) 'A unifying architecture for roadmaps frames a value scorecard', *Proceedings of the IEEE International Engineering Management Conference*, pp.383–386.

Bellomo, S., Nord, R.L. and Ozkaya, I. (2013) 'A study of enabling factors for rapid fielding combined practices to balance speed and stability', *35th International Conference on Software Engineering (ICSE)*, pp.982–991.

Bogsnes, B. (2008) *Implementing Beyond Budgeting: Unlocking the Performance Potential*, John Wiley & Sons, Hoboken, New Jersey.

Bryson, J.M. (2011) *Strategic Planning for Public and Nonprofit Organizations: A Guide to Strengthening and Sustaining Organizational Achievement*, Jossey-Bass, San Francisco, CA.

Butler, D. (2012) *Business Planning: A Guide to Business Start-Up*, Butterworth-Heinemann, Oxford.

Charette, R.N. (2003) *Challenging the Fundamental Notions of Software Development*, Cutter Consortium, Executive Rep, p.4, USA.

Cohn, M. (2006) *Agile Estimation and Planning*, Prentice Hall, NJ, US.

Cosner, R.R., Hynds, E.J., Fusfeld, A.R., Loweth, C.V., Scouten, C. and Albright, R. (2007) 'Integrating roadmapping into technical planning', *Research-Technology Management*, Vol. 50, No. 6, pp.31–48.

Creswell, J.W. (2003) *Research Design: Qualitative, Quantitative, and Mixed Method Approaches*, 2nd ed., Sage Publications, Thousand Oaks, California.

Easterbrook, S., Singer, J., Storey, M. and Damian, D. (2008) 'Selecting empirical methods for software engineering research', in Shull, F., Singer, J. and Sjøberg, D. (Eds.): *Guide to Advanced Empirical Software Engineering*, Springer, London.

Eppler, M.J. and Platts, K.W. (2009) 'Visual strategizing: the systematic use of visualization in the strategic-planning process', *Long Range Plann.*, Vol. 42, No. 1, pp.42–74.

Fitzgerald, B. and Stol, K. (2014) 'Continuous software engineering and beyond: trends and challenges', *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, pp.1–9.

Heikkilä, V.T., Paasivaara, M., Lassenius, C. and Engblom, C. (2013) *Continuous Release Planning in A Large-Scale Scrum Development Organization at Ericsson*, Springer, Berlin Heidelberg.

Highsmith, J. (2002) *Agile Software Development Ecosystems*, Addison-Wesley, Boston.

Hope, J. and Fraser, R. (2003) *Beyond Budgeting: How Managers Can Break Free from the Annual Performance Trap*, Harvard Business School Press, Boston, Massachusetts.

Järvinen, J., Huomo, T., Mikkonen, T. and Tyrväinen, P. (2014) 'From agile software development to mercury business', *Proceedings of the 5th International Conference (ICSOB)*, Springer, pp.58–71.

Järvinen, P. (2001) *On Research Methods*, Opinpajankirja, Tampere, Finland.

Kameoka, A., Kuwahara, T. and Li, M. (2003) 'Integrated strategy development: an integrated roadmapping approach', *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET)*, pp.370–379.

Kappel, T.A. (2001) 'Perspectives on roadmaps: how organisations talk about the future', *J. Prod. Innovation Manage.*, Vol. 18, No. 1, pp.39–50.

Kettunen, P. (2009) 'Adopting key lessons from agile manufacturing to agile software product development – a comparative study', *Technovation*, Vol. 29, No. 6, pp.408–422.

Koenigsaecker, G. (2009) *Leading the Lean Enterprise Transformation*, CRC Productivity Press Taylor & Francis Group, USA.

Kostoff, R.N. and Schaller, R.R. (2001) 'Science and technology roadmaps', *IEEE Trans. Eng. Manage.*, Vol. 48, No. 2, pp.132–143.

Leffingwell, D. (2007) *Scaling Software Agility: Best Practices for Large Enterprises*, Addison-Wesley, Boston, MA.

Leffingwell, D. (2011) *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, Pearson Education, Inc., Boston, MA, USA.

Lehtola, L., Kauppinen, M. and Vähäniitty, J. (2007) 'Strengthening the link between business decisions and RE: long-term product planning in software product companies', *15th IEEE International Requirements Engineering Conference (RE' 07)*, pp.153–162.

Lehtola, L., Kauppinen, M., Vähäniitty, J. and Komssi, M. (2009) 'Linking business and requirements engineering: is solution planning a missing activity in software product companies?', *Requirements Engineering*, Vol. 14, No. 2, pp.113–128.

Mavengere, N.B. (2013) 'Information technology role in supply chain's strategic agility', *International Journal of Agile Systems and Management*, Vol. 6, No. 1, pp.7–24.

Middleton, P. and Sutton, J. (2005) *Lean Software Strategies: Proven Techniques for Managers and Developers*, Productivity Press, New York.

Middleton, P., Flaxel, A. and Cookson, A. (2005) 'Lean software management case study: Timberline inc', *Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP)*, pp.1–9.

Myers, K.L. (1999) 'CPEF: a continuous planning and execution framework', *AI Magazine*, Vol. 20, No. 4, pp.63–69.

Naylor, J.B., Mohamed, M.N. and Berry, D. (1999) 'Leagility: integrating the lean and agile manufacturing paradigms in the total supply chain', *Int J Prod Econ*, Vol. 62, No. 1, pp.107–118.

Nordqvist, M. and Melin, L. (2010) 'The promise of the strategy as practice perspective for family business strategy research', *Journal of Family Business Strategy*, Vol. 1, No. 1, pp.15–25.

Olsson, H.H., Bosch, J. and Alahyari, H. (2013) 'Towards R&D as innovation experiment systems: a framework for moving beyond agile software development', *IASTED Multiconferences-Proceedings of the IASTED International Conference on Software Engineering, SE 2013*, pp.798–805.

Overby, E., Bharadwaj, A. and Sambamurthy, V. (2005) 'A framework for enterprise agility and the enabling role of digital options', *International Working Conference on Business Agility and Information Technology Diffusion*, Vol. 180, pp.295–312.

Petersen, K. (2010) 'Is lean agile and agile lean? A comparison between two software development paradigms', in Dogru, A.H. and Bicer, V. (Eds.): *Modern Software Engineering Concepts and Practices: Advanced Approaches*, IGI Global, USA.

Petersen, K. and Wohlin, C. (2010) 'Software process improvement through the lean measurement (SPI-LEAM) method', *J. Syst. Software*, Vol. 83, No. 7, pp.1275–1287.

Phaal, R. and Muller, G. (2009) 'An architectural framework for roadmapping: towards visual strategy', *Technological Forecasting and Social Change*, Vol. 76, No. 1, pp.39–49.

Phaal, R., Farrukh, C. and Probert, D. (2005) 'Developing a technology roadmapping system', *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET)*, pp.99–111.

Poppendieck, M. and Poppendieck, T. (2003) *Lean Software Development: An Agile Toolkit*, Addison-Wesley, Boston, MA.

Poppendieck, M. and Poppendieck, T. (2007) *Implementing Lean software development: From Concept to Cash*, Addison-Wesley Professional, Massachusetts, USA.

Radnor, Z. and Walley, P. (2008) 'Learning to walk before we try to run: adapting lean for the public sector', *Public Money and Management*, Vol. 28, No. 1, pp.13–20.

Rickards, R.C. and Ritsert, R. (2012) 'Rediscovering rolling planning: controller's roadmap for implementing rolling instruments in SMEs', *Procedia Economics and Finance of 2nd Annual International Conference on Accounting and Finance (AF 2012) and Qualitative and Quantitative Economics Research (QQE 2012)*, Vol. 2, pp.135–144.

Ruhe, G. (2010) *Product Release Planning: Methods, Tools and Applications*, CRC Press, USA.

Runeson, P. and Höst, M. (2009) 'Guidelines for conducting and reporting case study research in software engineering', *Empirical Software Engineering*, Vol. 14, No. 2, pp.131–164.

Shalloway, A., Beaver, G. and Trott, J.R. (2009) *Lean-Agile Software Development: Achieving Enterprise Agility*, Addison-Wesley Professional, Upper Saddle River, NJ.

Suomalainen, T., Salo, O., Abrahamsson, P. and Similä, J. (2011) 'Software product roadmapping in a volatile business environment', *J. Syst. Software*, Vol. 84, No. 6, pp.958–975.

Te Brömmelstroet, M. (2013) 'Performance of planning support systems: what is it, and how do we report on it?', *Comput., Environ. Urban Syst.*, September, Vol. 41, pp.299–308.

Van de Weerd, I., Bekkers, W. and Brinkkemper, S. (2010) 'Developing a maturity matrix for software product management', *Proceedings of the 1st International Conference on Software Business (ICSOB 2010)*, pp.76–89.

Van Oosterhout, M., Waarts, E. and van Hillegersberg, J. (2005) 'Assessing business agility: a multi-industry study in the Netherlands', *Business Agility and Information Technology Diffusion*, Vol. 180, pp.275–294.

Wareham, T.L. and Majka, A.J. (2003) *Best Practice Financing*, Kaufman Hall White Paper, Kaufman, Hall & Associates, Northfield, IL.

Westkamper, E. and von Briel, R. (2001) 'Continuous improvement and participative factory planning by computer systems', *CIRP Annals-Manufacturing Technology*, Vol. 50, No. 1, pp.347–352.

Womack, J.P. and Jones, D.T. (2003) *Lean Thinking: Banish Waste and Create Wealth in Your Corporation, Revised and Updated*, Free Press, USA.

Yin, R.K. (1994) *Applied Social Research Methods Series Vol 5; Case Study Research: Design and Methods*, 2nd ed. Sage Publications, Inc. Yin, R.K, Thousand Oaks, California.

PAPER IV

Defining Continuous Planning through a Multiple-Case Study

Proceedings of the 16th International Conference of Product-Focused Software Process Improvement (PROFES). LNCS 9459, pp. 288–294.
Bolzano, Italy, December 2–4, 2015.

PAPER V

Continuous deployment of software intensive products and services: A systematic mapping study

Journal of Systems and Software, in press.

The Journal of Systems and Software 000 (2016) 1–29

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Continuous deployment of software intensive products and services: A systematic mapping study

Pilar Rodríguez[a,*], Alireza Haghighatkhah[a], Lucy Ellen Lwakatare[a], Susanna Teppola[b], Tanja Suomalainen[b], Juho Eskeli[b], Teemu Karvonen[a], Pasi Kuvaja[a], June M. Verner[c], Markku Oivo[a]

[a] Department of Information Processing Sciences, 90014 University of Oulu, Box 3000, Finland
[b] VTT Technical Research Centre of Finland, P.O. Box 1100, FI-90571 Oulu, Finland
[c] Keele University, UK

ARTICLE INFO

Article history:
Received 17 January 2015
Revised 15 October 2015
Accepted 7 December 2015
Available online xxx

Keywords:
Continuous deployment
Software development
Systematic mapping study

ABSTRACT

The software intensive industry is moving towards the adoption of a value-driven and adaptive real-time business paradigm. The traditional view of software as an item that evolves through releases every few months is being replaced by the continuous evolution of software functionality. This study aims to classify and analyse the literature related to continuous deployment in the software domain in order to scope the phenomenon, provide an overview of the state-of-the-art, investigate the scientific evidence in the reported results and identify areas suitable for further research. We conducted a systematic mapping study and classified the continuous deployment literature. The benefits and challenges related to continuous deployment were also analysed. RESULTS: The systematic mapping study includes 50 primary studies published between 2001 and 2014. An in-depth analysis of the primary studies revealed ten recurrent themes that characterize continuous deployment and provide researchers with directions for future work. In addition, a set of benefits and challenges of which practitioners may take advantage were identified. CONCLUSION: Overall, although the topic area is very promising, it is still in its infancy, thus offering a plethora of new opportunities for both researchers and software intensive companies.

1. Introduction

The software intensive industry is evolving towards a value-driven and adaptive real-time business paradigm (Järvinen et al., 2014). *The Age of Information* (Castells, 2011), which is strongly based on the *Internet speed-of-things*, has shaped a digital economy and a knowledge-based society. Digital resources are constantly available for everyone, information flows are accelerated and individuals can explore their personal needs more easily. Consequently, fast-changing and unpredictable markets have shifted the competitive software development landscape. The traditional view of software as a static item that can be bought and owned is giving way to software services in which customers expect a continuous evolution of product functionality that provides additional value (Bosch, 2012). These market features enable new business opportunities. However, they also exert pressure to develop *dynamic capabilities* (Eisenhardt and Martin, 2000). To maintain their competitive advantage, software intensive companies need to deliver valuable product features to customers considerably faster than before, if not near to real-time, while embracing business changes and pursuing economic efficiency.

Agile software development (ASD) emerged in 2001 (Beck et al., 2001) as a ground breaking foundation for new software development processes. Iterative development, continuous integration and short feedback cycles were advocated as a replacement for the traditional engineering stage-gate models. The ultimate aim of ASD was to improve the organization's capability to adapt to market fluctuations and customer needs. Although ASD was initially considered a fad, and caused some controversies (Boehm, 2002), it became progressively mainstream. Software practitioners have increasingly adopted ASD (Rodríguez et al., 2012), and the research in the area has become well-established (Dingsøyr et al., 2012).

A recent evolutionary step from agile and lean software development is rapid and continuous software engineering. Rapid and continuous software engineering refers to the *'organizational capability to develop, release and learn from software in rapid parallel cycles, such as hours, days or very few weeks'* (ICSE 2014, http://continuous-se.org/). ASD is extended to approaches where the

* Corresponding author. Tel.: +358 40 1602 179.
E-mail address: pilar.rodriguez@oulu.fi (P. Rodríguez).

http://dx.doi.org/10.1016/j.jss.2015.12.015
0164-1212/

step between development and deployment is minimalized in order to deploy code immediately to production environment for customers to use. Continuous deployment (CD) is the term used to refer to this phenomenon (Humble and Farley, 2010; Olsson et al., 2012; Fitzgerald and Stol, 2014; Järvinen et al., 2014; Claps et al., 2015). Although the concept of deploying software to customers as soon as new code is developed is not new and is based on ASD principles, CD extends ASD by moving from cyclic to continuous value delivery. This evolution requires not only agile processes at the team level but also integration of the complete R&D organization, parallelization and automation of processes that are sequential in the value chain and constant customer feedback.

Leading organizations, such as Facebook, Microsoft and IBM, have provided examples of CD implementation (Claps et al., 2015), which has led to the emergence of studies related to CD in the scientific literature (e.g. Olsson et al., 2012; Cukier, 2013; Feitelson et al., 2013; Krusche et al., 2014). Mäntylä et al. (2014) recently published a semi-systematic literature review as part of their research on rapid releases and testing. Although their findings are significant and shed light on the grey area of CD, the CD research field remains dispersed among different research areas and a structured understanding of the main factors that characterize CD is not provided. Therefore, the goal of this study is to identify the state of the art of the phenomenon of CD in the context of software development. This systematic mapping study (Kitchenham et al., 2011) aims at the following:

1. To establish the body of knowledge of CD by identifying and categorizing the available research on the topic
2. To assess the quality of the existing research in terms of industrial relevance and research rigour
3. To identify the most relevant articles in the field of CD
4. To determine the underlying factors that characterize CD as both a concept and a phenomenon
5. To provide baselines to assist with further research

This study combines the process of a systematic literature review (SLR), as established by Kitchenham and Charters (2007), with the mapping study, as suggested by Petersen et al. (2008). We wish to present a logical organization of the CD literature in order to provide researchers with a structured body of knowledge on which to base their studies. We also want to furnish practitioners with the main factors they should consider when deciding whether or not to migrate to CD, the benefits that they can expect, as well as the potential risks and challenges they might face with CD.

The remainder of this paper is organized as follows: background and related work are presented in Section 2. Section 3 describes the research methodology, including a discussion on threats to validity and countermeasures taken to minimize their effects. In Section 4, we present the results of the mapping study. Sections 5 and 6 provide an analysis of the factors, benefits and challenges characterizing CD. Opportunities for future research are discussed in Section 7. Section 8 presents a comparison of our findings with related work and, concretely, to the semi-systematic literature review conducted by Mäntylä et al. (2014). Finally, we present our conclusions in Section 9.

2. Background and related work

The roots of CD began fifteen years ago with the formulation of the Agile Manifesto (Beck et al., 2001). Some argued that ASD was just *old wine in new bottles* (Merisalo-Rantanen et al., 2005), in reference to the roots of ASD in iterative models, such as the spiral model (Boehm, 1988). Undoubtedly, today's software engineering is the result of the evolutions of previous software development models (Boehm, 2006). This section describes the phenomenon of CD, providing a historical view and identifying its core ideas. We then summarize previous literature reviews related to CD and justify the need for this review and its research questions.

2.1. Continuous deployment

The *Internet-speed of things* has changed the way software companies deliver value to their customers. The widespread adoption of lean principles and agile methodologies (Rodríguez et al., 2012) has provided evidence of the need for and value of flexibility and adaptation in the current environment of software development (Fitzgerald and Stol, 2014). ASD established some foundations for CD. For example, agile principles, such as *'our highest priority is to satisfy the customer through early and continuous delivery of valuable software'* and *'deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale'* make clear reference to CD. However, ASD has mainly focused on speeding up the development process at the team level through methods such as eXtreme Programming (Beck, 2000) and Scrum (Schwaber, 2004). CD moves beyond the concept of ASD towards a situation in which software functionality is continuously deployed to the final customers (production environment), and where customer input is the main driver for innovation (Olsson et al., 2013). Instead of working for months on a major new release, companies limit their cycle time (i.e., the time between two subsequent releases) to a couple of weeks, days or, in some cases, even hours (Mäntylä et al., 2014). A plethora of evidence related to CD exists in organizational white papers and online blogs, where practitioners have voiced their experiences and expectations in moving to CD (e.g. Facebook,[1] IBM,[2] Microsoft,[3] Google,[4] Adobe,[5] Netflix[6] and IMVU[7]). Accordingly, a body of academic literature is also emerging on this topic.

Humble and Farley (2010), in their book on CD published in 2010, state that continuous delivery provides enterprises with the ability to deliver rapidly, reliably and repeatedly value to customers at low risk with minimal manual overhead. While continuous integration, which is a core ASD practice, mainly focuses on the automation of the build process (the code is built, and a set of unit tests are run when it is checked into version control repository), continuous delivery is a logical progression that automates the entire workflow simplifying the rapid release of software. The central concept in Humble and Farley's approach is a deployment pipeline that establishes an automated end-to-end process to ensure that the system works at technical level, executes fairly automated acceptance tests and lastly deploys to a production or staging environment.

The concepts underlying CD have also attracted the attention of researchers. Recent editions of the ICSE[8] conference (2013, 2014, 2015) have offered workshops focused on the topic such as the International Workshop on Release Engineering (RELENG), which emphasized *'the recent trend to reduce the release cycle to days or even hours'*, as well as the International Workshop on Rapid and Continuous Software Engineering (RCoSE). Drawing upon the concept of rapid and continuous software engineering, Fitzgerald and Stol (2014) propose a conceptual model that presents a set of con-

[1] https://www.facebook.com/notes/facebook-engineering/ship-early-and-ship-twice-as-often/10150985860363920 (accessed 09, 2015).
[2] http://www.ibm.com/developerworks/rational/library/continuous-deployment-rational-alm/ (accessed 09, 2015).
[3] http://blogs.msdn.com/b/bharry/archive/2012/06/07/announcing-continuous-deployment-to-azure-with-team-foundation-service.aspx (accessed 09, 2015).
[4] https://air.mozilla.org/continuous-delivery-at-google/ (accessed 09, 2015).
[5] http://steveblank.com/2014/01/06/15756/ (accessed 09, 2015).
[6] http://steveblank.com/2014/01/06/15756/ (accessed 09, 2015).
[7] http://timothyfitz.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/ (accessed 09, 2015).
[8] International Conference on Software Engineering, http://icse-conferences.org/ .

tinuous activities across the software development lifecycle with a holistic view of business, development, operations and innovation activities. In a similar vein, Järvinen et al. (2014) propose a model to characterize the evolution of enterprises towards and beyond real-time value delivery. The model is based on concepts borrowed from the *Elastic Enterprise* (Vitalari and Shaughnessy, 2012) and the *Lean Startup* framework (Ries, 2011). The authors argued that in the current economic climate, software companies require new capabilities to: (1) deliver value in real-time; (2) increase customer insight through active customer involvement and rapid experimentation; and (3) seek new ways of executing their existing businesses to enable them to move into completely new business areas when needed. The last stage is called *mercury business*.

According to Olsson et al.'s model (Olsson et al., 2013), companies evolve from traditional development to ASD through the careful adoption of agile practices and a shift to smaller cross-functional teams. When an organization matures in the use of agile, and uses automated system integration and verification, then that organization can take the next step, which is the adoption of continuous integration. When continuous integration is in place, customers often express an interest in receiving enhancements and bug fixes more frequently, so the organization migrates to CD. The final step occurs when the organization not only releases software continuously but also develops mechanisms to conduct rapid experimentation in order to drive innovation.

Although similarities and differences exist among the emerging models of CD, three major themes characterize the models: (1) deployment, (2) continuity and (3) speed. Hence, continuous deployment means the ability to bring valuable product features to customers on demand and at will (deployment), in series or patterns with the aim of achieving continuous flow (continuity) and in significantly shorter cycles than traditional lead-times, from a couple of weeks, to days or even hours (speed). In addition, some authors distinguish between delivery and deployment. For example, Humble and Farley (2010) describe continuous deployment as the automatic deployment of every change to production, whilst continues delivery is an organizational capability that ensures that every change can be deployed to production, if it is targeted (however, the organization may choose not to do it, usually due to business reasons). However, most of the scientific literature uses the terms continuous deployment and continuous delivery interchangeably.

2.2. Related work

This section clarifies the need for our CD study. Several studies have systematically analysed the literature on areas related to CD. However, no review has specifically focused on actually structuring the body of CD knowledge (Zhang and Ali Babar, 2013).

ASD and its practices have been the topics of diverse SLRs. Dybå and Dingsøyr (2008) conducted a well-known SLR on empirical studies of ASD published up to and including 2005. In this review, 33 relevant primary studies were identified and classified into four thematic groups: introduction and adoption, human and social factors, perceptions of agile methods and comparative studies. They found a steady increase in the number of studies on ASD. However, they also observed poor quality with regard to the research methods used in most studies, and they suggested an increase in both the number and the quality of empirical studies on ASD. However, the focus in Dybå and Dingsoyr (2008) is different from a review of CD as most primary studies included focus on agile methods at the team level instead of integration of the complete R&D organization. As noted by Olsson et al. (2013), transition to CD involves evolving agile practices beyond the R&D organization to ensure that other functions such as product management and sales are functioning in an agile manner as well.

As the body of knowledge on ASD has matured, other authors focused their analyses on specific ASD practices. Test-driven development (TDD) is one of the most reviewed areas (Causevic et al., 2011; Turhan et al., 2010). However, the described benefits and drawbacks of TDD were inconclusive. Similarly, continuous integration was the subject of several SLRs (Ståhl and Bosch, 2014; Eck et al., 2014). Ståhl and Bosch (2014) performed a SLR in order to investigate how the practice of continuous integration was implemented in practice. Twenty-two descriptive themes, including build duration, build frequency, and pre-integration procedures were extracted from 46 publications. The authors concluded that a multitude of continuous integration variants exists in practice, but there was no consensus on continuous integration as a single homogeneous practice. Similarly, Eck et al. (2014) conducted a SLR to examine how organizations assimilate continuous integration. Based on 43 studies, the authors presented a conceptual framework illustrating the assimilation stages of continuous integration, which included acceptance, routinization and infusion. Again, although continuous integration is a key aspect of CD (as our findings also confirm), continuous integration merely focuses on automating the build process. In general, although ASD and its practices are within the scope of this research, because they represent the origins of CD (Mäntylä et al., 2014), we are interested in ASD only as far as it explicitly supports CD.

Recently, Mäntylä et al. (2014) published a semi-systematic literature review as part of a study focused on rapid releases and testing. They analysed the current tendency towards rapid releases, as well as its benefits, enablers and problems. This review showed that rapid release is a prevalent practice in industry that originated with the agile, open source, lean and Internet speed development movements. Parallel development, strong tool infrastructure for automatic deployment and testing, as well as pro-active customers and product managers, were found to be enablers of rapid release, whilst time pressure, increased technical debt, customer un-willingness to update, as well as conflicting goals between rapid release and achieving high reliability and test coverage, were found to potentially cause problems. In addition, shorter time-to-market, rapid feedback, customer satisfaction, increased efficiency, improved quality focus and easier monitoring of progress and quality were identified as benefits. Although Mäntylä et al.'s study sheds light on the grey area of CD, what characterizes and constitutes the phenomenon of rapid releases was unclear.

In summary, CD has attracted a lot of interest from the software industry within a short time, and the notion of CD among practitioners is growing. In addition, research related to CD is emerging in the scientific literature. However, the literature is not well structured, and there is no clear understanding of the main factors that characterize CD. The existing literature reviews, which were presented in the previous section (i.e., ASD literature reviews and Mäntylä et al.'s semi-systematic literature review (Mäntylä et al., 2014)), only partially cover the existing scientific studies on CD. Furthermore, the validity of the evidence in the CD literature is unclear because no previous review evaluated the quality of the published studies.

3. Research methodology

A systematic mapping study was conducted to obtain an overview of the research on CD. The main difference between systematic mapping studies and SLRs is that while SLRs aim to *'identify best practice with respect to specific procedures, technologies, methods or tools by aggregating information from comparative studies'*, mapping studies focus on *'classification and thematic analysis of*

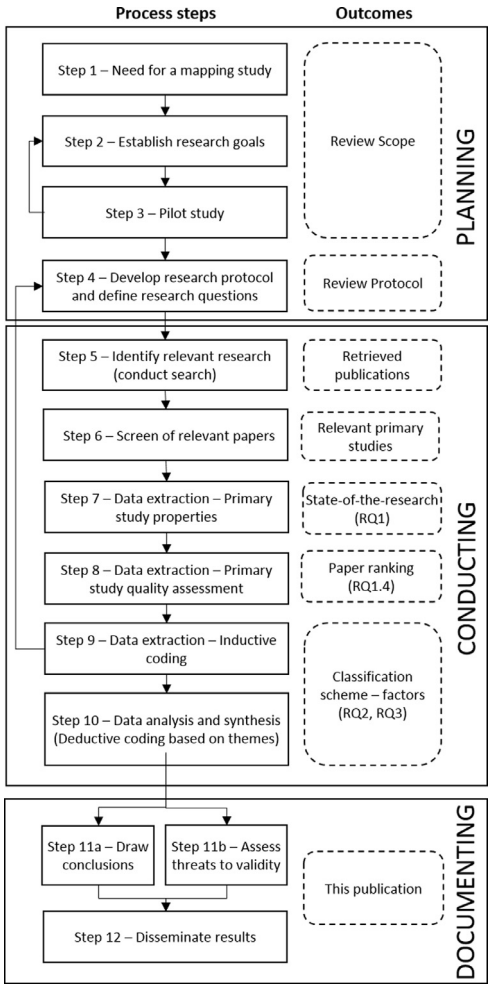


Fig. 1. Mapping study steps.

literature on a software engineering topic' (Kitchenham et al., 2011).[9] In the case of CD, although the term is frequently used in industrial and academic circles (see Section 2), its meaning and the main factors that are part of CD have remained undefined. Therefore, before aggregating information in terms of research outcomes, we need to provide a comprehensive definition of CD, that is, identify, categorize and analyse the available research on the topic of CD in order to describe the phenomenon, obtain an overview of its state-of-the-art, determine the scientific evidence in the reported results and determine areas that are suitable for more detailed study.

In our mapping study, we followed the process of a SLR as established by Kitchenham and Charters (2007), but we adapted it to a mapping study, as suggested by Petersen et al. (2008). The research process is outlined in Fig. 1. The process was iterative, with feedback loops between the steps that helped to focus the mapping study as we learned more about the phenomenon itself. For example, the search string was piloted until we found one that ensured the most complete access to the body of knowledge about CD. Furthermore, additional research questions were incorporated when we learned more about the data available in the primary studies. However, in order to achieve a legible description of how the research was conducted, the research process is described in a sequential manner, emphasizing the important design decisions taken during the process.

This section describes the design and execution of our systematic mapping study. The complete research package, which includes the research protocol that helped maintain the chain of evidence and the transparency between each step in the process, is available on request.

3.1. The need for the study and definition of research questions

The need for this mapping study (step 1, Fig. 1) emerged in the context of the large Finnish research programme, *Need for Speed* (N4S, 85 M€ , 2014–2017).[10] The main objective of this study (step 2, Fig. 1), which is mentioned in Section 1, is to identify the relevant CD literature in order to: (1) create a knowledge base in the area; (2) understand the factors that characterize the nature of CD in the software intensive industry; (3) critically determinate the scientific evidence reported in the CD literature; and (4) identify areas that should be addressed in future research. Our objectives are expressed in the form of the research questions presented in Table 1 (step 4, Fig. 1). The research questions (RQ) were defined from a broad perspective. RQ1 concerns the state of the research pertaining to CD; RQ2 concerns the main factors of CD; and RQ4 concerns the research gaps in the area of CD. After reading the entire set of primary studies and gaining a better understanding of the data (step 9, Fig. 1), we discovered that different definitions of CD were present in the literature, as well as benefits and challenges regarding its adoption. Accordantly, RQ2 was supplemented with RQ2.1 and RQ3 was formulated to complete the set of research questions. The research questions defined the data to be extracted from the primary studies (see Section 3.5).

3.2. Search strategy and databases

Several experimental searches were piloted from April to June 2014 (Fig. 1, step 3) in order to better scope the research and determine the search string that most appropriately describes the phenomenon. The following search string was the first used in the review:

Pilot search string: *("continuous delivery" OR "continuous deployment") AND "software"*

Using this string, 28 hits were found in Scopus, and 17 of the 28 were closely studied. Eleven were excluded because they were duplicated, did not focus on the software domain or were non-scientific. Based on an initial reading of the 17 studies, the search string was revised to include the keywords that best described the phenomenon of CD. Table 2 presents the final search string used to retrieve the primary studies together with the rational for the selection of those terms. The search string is composed of terms that represent the population AND intervention, as proposed by Kitchenham and Charters (2007). Our research focus is on scientific studies that discuss software (population) AND have the intention of deployment [OR closely related terms] in a continuous [OR closely related terms] manner (intervention).

To increase publication coverage, we decided to use the broad term *software* to ensure that we would not miss relevant references. In addition, during the pilot study, we discovered that various terms referred to the concept of continuous, such as real-time or rapid; therefore, we added these terms to the search string. We also learned that relevant studies, such as Eklund and Bosch (2012) and Humble et al. (2006) were missed when the terms *continuous* and *delivery* or *deployment* were used together (e.g. 'continuous deployment'). Therefore, we decided to allow the term *continuous* (or its closely related terms) to be separated from the terms *deployment, delivery* and *release*. These design decisions, together with

[9] A more detailed description of when to use systematic mapping studies and a discussion of its main differences from SLRs is presented by Kitchenham et al. (2011).

[10] http://www.n4s.fi/en/.

Table 1
Research questions for the mapping study.

ID	Question	Aim
RQ1	*What is the current state of the research pertaining to CD in the context of software intensive products and services?*	Providing an overview of the studies on CD in the context of software intensive products and services.
RQ1.1	What research methods have been used in studies related to CD?	To categorize available CD research according to research type (industry report, case study, experiment, survey questionnaire, theory development, etc.).
RQ1.2	What kinds of contributions are provided by studies related to CD?	To categorize available CD research according to its contribution facet (model, theory, framework/method, guidelines, lessons learned, advice/implication or tool).
RQ1.3	What are the publication channels used to publish studies related to CD?	To gain an overview of the publication channels used for CD studies (conference or journal) and publication years' frequency distribution.
RQ1.4	What are the levels of relevance and rigour in the published articles?	To assess the quality of the CD studies by examining two perspectives: their industrial relevance and scientific rigour.
RQ2	*What are the main factors that characterize CD in the context of software intensive products and services?*	To structure the state of the art of CD by identifying and analysing the underlying themes associated to this phenomenon discussed in the literature and that, therefore, define it.
RQ2.1	What do researchers mean when they refer to the term CD in the context of software intensive products and services?	To identify and analyse the different definitions of CD available in the literature.
RQ3	*What are the reported benefits and challenges in association with CD in the context of software intensive products and services?*	To identify the reported benefits and challenges experienced when using CD.
RQ4	*What are the research gaps in the area of CD in the context of software intensive products and services?*	To identify research gaps in the field of CD and opportunities for further research.

Table 2
Search keywords.

	Search term	Rational
Population	software	Studies discussing software, software development, software engineering or software intensive products/services/systems.
AND		
Intervention	deploy*	Studies discussing "deployment" in the context of software (e.g. deploy, deployment, deploying).
	deliver*	Studies discussing "delivery" in the context of software (e.g. deliver, delivery, delivering).
	releas*	Studies discussing "releas" in the context of software (e.g. release, releasing).
	AND	
	continuous*	Studies discussing "continuous" in the context of software delivery or deployment (e.g. continuous, continuously).
	rapid*	Rapid is a closely related term to "continuous" (e.g. rapid, rapidly).
	fast*	Fast is a closely related term to "continuous" (e.g. fast, faster).
	real time	Real-time is a closely related term to "continuous" (e.g. "real-time delivery").
	agil*	Agile is a closely related term to "continuous" (e.g. agile, agility).
	iterat*	Iterative is a closely related term to "continuous" (e.g. iterative, iteration, iterations).
	increment*	Incremental is a closely related term to "continuous" (e.g. incrementally, increment, incremental).

the extensive usage of closely related terms, increased the number of studies retrieved (i.e., introduced noise in the search) but reduced the risk of missing relevant studies. We used this search string to search within keywords, titles and abstracts.

The selected databases in which we performed the search are shown in Table 3, in addition to the number of studies retrieved from each database (up to and including June 2014).[11] The databases were selected considering their coverage of the software engineering literature. The individual search strings in the study protocol are available by request.

Table 3
Selected databases and retrieved papers.

Database	Filter	Papers
ACM Digital Library	None	933
Scopus	Only conference papers and journal articles in English in the following subject areas: computer science, engineering, business management and accounting	7997
IEEE Xplore	Only conference papers and journal articles	5303
ISI Web of Science	Only articles in the following research areas: engineering, computer science and telecommunications	5215
Science Direct	Only conference papers and journal articles	1934
		Total 21,382

3.3. Primary study selection criteria

Studies were eligible for inclusion in the mapping study if they presented a scientific contribution to the body of CD knowledge in the context of the software-intensive industry. Concretely, the inclusion criteria were defined as *'any study that is a scientific article and clearly states that it focuses on software development or software intensive products, systems or services AND includes any software development activity as primary subject with the intention of continuous deployment or delivery of a software product, system or service'*.

Because our goal was to analyse trends in the area of CD rather than aggregate empirical evidence gathered from individual studies, both theoretical and empirical studies were included in the mapping study (Kitchenham et al., 2011). Similarly, studies conducted in both industry and in academia were included. Three

[11] Performed on 27 June 2014.

aspects were considered during the screening process used to evaluate whether the content of an article was relevant to CD: deployment, continuity and speed. To be included in the review, the study had to include the following: (1) show intention/ability of bringing a software product/system/service to the production environment in order to be used by the customer (deployment); (2) emphasize continuous series or patterns so that the software is deployed repeatedly, providing a continuous evolution of software functionality (continuity); and (3) focus on significantly shorter cycles than traditional development lead-times, preferably near to real time, at will or on-demand (speed).

We excluded search results that:

1. Did not clearly discuss the continuous deployment or delivery of software intensive product/systems/services.
2. Were not related to the software domain (e.g., medicine, biology, physics, etc.).
3. Were not peer-reviewed scientific articles (e.g. presentations, call for papers, keynote speeches, prefaces, etc.) or were book and book chapters.
4. Were short papers.
5. Were not written in English.
6. Were duplicate articles.

For example, because the focus of this mapping study is on the phenomenon of CD, general discussions about ASD, lean software development or any of their practices and tools without an explicit application to CD were excluded. In addition, we imposed no limitations with regard to quality (rigour and relevance) in selecting primary studies.

3.4. Primary study selection procedure

To screen the retrieved publications (steps 5 and 6, Fig. 1) we followed the process shown in Fig. 2. Due to the high number of publications found (21,382) and the inconsistency between the meta-data format stored in different databases, we decided to use the reference management system RefWorks[12] that automated the task of aggregating papers into a consistent list of candidate papers in a unified format. The selection procedure comprised the following steps:

First, two researchers together went through the list of 21,382 candidate publications in order to eliminate duplicate, non-English publications, non-relevant software engineering studies and non-peer review scientific articles (exclusion criteria 2–6). Non-relevant software engineering studies were identified by checking the publication forum and the publication title. Obviously non-scientific peer review publications, such as those titled *'A call for …'* or *'Proceedings of …'* as well as introductions of workshops and editorials were also identified and removed. At the end of this stage, the number of remaining papers was 9924. In the second stage, based on exclusion criterion 1, we screened candidate papers by conducting a conservative in-depth review. We excluded papers only when it was clear that they were not within the scope of our research. In unclear cases, the paper was passed to the next screening phase. First, two researchers (simultaneously) read the titles of the remaining papers. Generally, it was difficult to identify whether the focus of the study was on CD based only on its title. Therefore, papers that clearly did not focus on CD were excluded in this step; 7217 papers were excluded. Then two researchers (individually) went through the list of remaining publications (2707) to screen their abstracts. When they crosschecked the outcome, they agreed to exclude 2377 papers and include 16 papers. However, based on abstracts they could not decide on 314 papers, which were categorized as *'unsure'*. Introduction, conclusions an, when needed, the whole paper was read in order to resolve *'unsure'* cases. In cases of conflict between two researchers a third researcher helped to resolve the conflict. Finally, 34 studies were added to the list of the 16 papers already included (based on title). Therefore, 50 papers remained in the final pool of primary studies. The primary studies (PS) are included in the reference list at the end of this paper (identified by the symbol *[PS]).

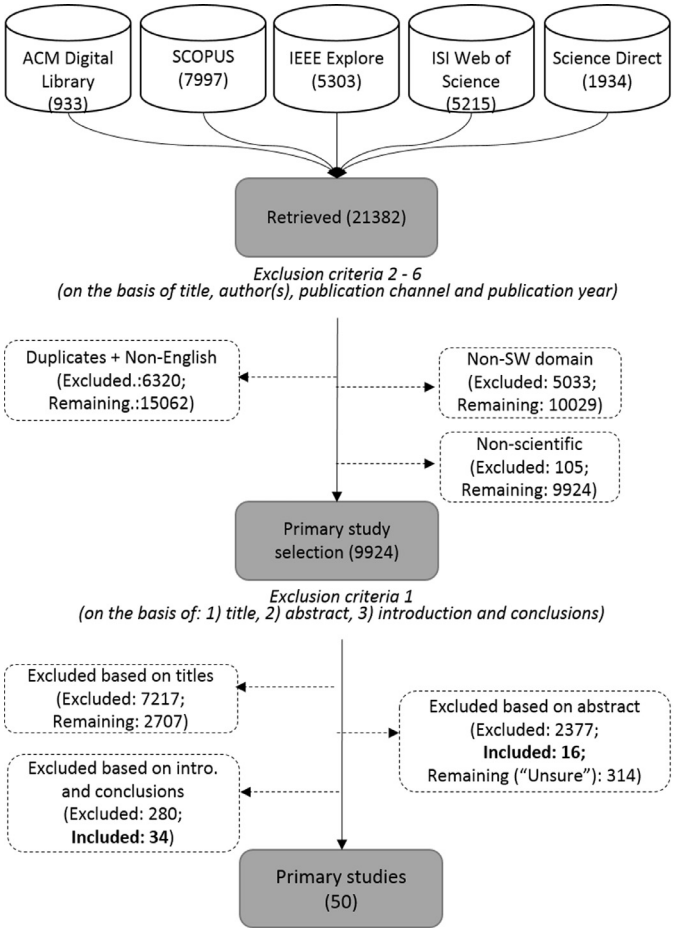


Fig. 2. Screening of papers.

Table 4
Data extraction form.

ID	Property	Research question(s)
P1	General type of paper	RQ1.1
P2	Research method	RQ1.1
P3	Contribution	RQ1.2
P4	Domain	RQ1.2
P5	Pertinence	RQ1.2
P6	Publication year and forum	RQ1.3
P7	Quality – rigour	RQ1.4
P8	Quality – relevance	RQ1.4
P9	CD factor	RQ2
P10	CD definition	RQ2.1
P11	Benefit from using CD	RQ3
P12	Challenge for adopting CD	RQ3

3.5. Data extraction

Table 4 lists the properties collected during the data extraction phase. Based on the RQs and using the qualitative analysis tool NVivo,[13] three categories of data were extracted from the primary

[12] http://www.refworks.com/.

[13] http://www.qsrinternational.com/.

studies: (1) primary study properties (P1–P6); (2) quality assessment information (P7 and P8); and (3) recurrent themes (P9–P12). The next subsections describe each extracted property.

3.5.1. Primary study properties: P1–P6

Six primary study properties, which are briefly described below, were defined in order to answer RQ1.1–RQ1.3. The detailed definitions of these categories are presented in Table A.9, Appendix A.

1. *General type of paper*: represents the type of research (empirical, theoretical or both). Definitions of these values were adapted from Kitchenham (2010).
2. *Research method*: categorizes the studies according to the applied research method. Ten categories were considered: case study, industry report, experiment, survey, action research, mixed methods, grounded theory, design science, opinion paper and not stated. When possible, we used the definitions provided by Unterkalmsteiner et al. (2012). Otherwise we created our own definition.[14]
3. *Contribution*: maps different types of study outcomes (inspired by Paternoster et al., 2014). Seven categories of contribution (adapted from Shaw, 2003) were chosen: model, theory, framework or method, guidelines, lessons learned, advice or implications, and tools.
4. *Domain*: maps the different types of domain in which CD is used. Four categories were used to classify the domains of the primary studies: 'embedded systems', 'web/internet based applications or services', 'desktop applications' and 'not stated/not clear' (to indicate that the domain was not clearly stated in the paper).
5. *Pertinence*: this property (inspired by Paternoster et al., 2014) was designed to distinguish between studies entirely devoted to CD and studies with a broader perspective. The values of pertinence were defined as fully, partially or marginally focused on CD.
6. *Publication year and channel*: categorizes primary studies according to publication channel (conference or journal) and provides the frequency distribution of the publication years.

3.5.2. Primary study quality assessment: P7 and P8

Primary study quality assessment (e.g. evaluating how reliable and relevant the study is) is critical in SLRs because the aim of SLRs is to aggregate the results of primary studies in order to discover whether the research outcomes are consistent or contradictory. Therefore, ensuring that results are comparable and based on the best evidence is critical in SLRs. However, analyzing the quality of the primary studies is not essential in the case of mapping studies, which aim to classify the relevant literature (Kitchenham et al., 2011). In our case, we analyzed the quality or our primary studies as one characteristic of the state of the art of CD (RQ1.4). Therefore, the primary studies were not filtered (excluded) based on their quality but all primary studies were considered in the analysis phase independently of the quality assessment results (i.e., categorizing the topic area according to their meta-data [RQ1], counting the number of studies in those categories and identifying and analysing the factors that characterize CD [RQ2 and RQ3]).

To assess the quality of our primary studies (Fig. 1, step 8), we applied the method proposed by Ivarsson and Gorschek (2011). Accordingly, we considered two perspectives: scientific rigour and industrial relevance. Three aspects were used to evaluate scientific rigour: (1) context description: to what degree the context of the study is described so that it can be understood and compared to another context, allowing the replication of the study; (2) study design: to what degree the design of the study is properly described and guarantees the rigour of the research; and (3) validity discussion: to what extent the validity of the study is considered and evaluated. Rigour was evaluated using a three-point scale: strong description (1), medium description (0.5) and weak description (0). Thus, the assessment of rigour ranged from 3 to 0. On the other hand, industrial relevance was evaluated according to four aspects: (1) subjects: whether the subjects of the study were representative of CD practitioners (i.e., they were not students or researchers); (2) context: whether the study was performed in a representative setting (i.e., industrial setting); (3) scale: whether the study size was realistic (i.e., not based on a 'toy' example); and (4) research method: whether the research method used in the study contributes to an investigation of real situations (typically empirical research conducted in representative settings contributes in this sense). In accordance to Ivarsson and Gorschek's method, relevance was measured using two values: 1 if the aspect contributed to industrial relevance and 0 otherwise. Therefore, the assessments of industrial relevance ranged from 4 to 0. To analyse the quality of theoretical papers, we adapted Ivarsson and Gorschek's model (Ivarsson and Gorschek, 2011) to the characteristics of theoretical studies. Industrial relevance was measured in the same way. Therefore, most theoretical papers had low industrial relevance if they did not include any empirical evaluation of their proposed model, framework or tool. The criteria for rigour were adapted as follows: (1) context description: description of the context in which the theory or model could be applied; (2) study design: the degree to which the theoretical contribution used sound theoretical bases to guarantee the quality of the research; and (3) validity discussion: the extent to which the limitations of the theoretical approach were discussed. For a detailed discussion of the criteria used in Ivarsson and Gorschek's model, the reader is referred to their extensive study on this subject (Ivarsson and Gorschek, 2011).

3.5.3. Continuous deployment factors: P9 - P12

We used the concept *'factor'* (P9) to identify and categorize the recurrent themes in the literature related to CD and create our classification schema (RQ2). During the primary study coding process (see Section 3.6), we identified factors defining aspects that, according to our primary studies, are relevant in achieving CD. That is, aspects that are frequently considered in the literature in order to implement CD in practice. The classification schema grew with the extraction of the data from the primary studies, and it was consolidated in workshops attended by the first seven authors. Similarly, we identified diverse definitions of CD (P10) and compared them in order to understand what researchers meant when they refer to the concept of CD (RQ2.1). Finally, in order to answer RQ3, we coded the benefits that the studies claimed were gained from the use of CD (P11), as well as challenges or aspects that were difficult to implement in the context of CD (P12).

3.6. Data analysis and interpretation

Descriptive statistics were used to answer RQ1. Quantitative descriptions of frequencies were used to analyse research methods, types of contributions, publication channels, publication years, and quality of primary studies.

In addition, thematic synthesis (Cruzes and Dyba, 2011) was used to answer RQ2 and RQ3. In order to identify CD factors and create a reliable classification schema, we analysed each primary study in two phases, adapting the five thematic synthesis steps recommended by Cruzes and Dybå (2011). Both inductive and deductive coding were used in the analysis. First, primary studies were coded using an inductive approach (Fig. 1, step 9). The goal of this step was to

[14] For those research methods that (Unterkalmsteiner et al., 2012) did not consider in their classification.

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

identify key CD aspects. Codes emerged from labelling relevant segments of text that referred to factors important in the context of CD. For example, many papers made reference to *automating the deployment* and a correspondent code was created. Each primary study was coded by one researcher. Then two full-day workshops were conducted in order to review the generated codes and the coding process itself. After a stable list of *'free-codes'* was created, the codes of all primary studies were compared and then organized into higher-order interpretation themes in order to form a high-level set of categories (descriptive themes). For example, we found codes that referred to automating different activities in the development process such as *test automation, build automation, integration automation, deployment automation, automation and configuration management*, and we created the theme *automation* in order to consider this aspect of CD. Two full-day workshops were held in order to translate the codes into themes. When the themes were defined, the second coding phase, which was deductive, was implemented (Fig. 2, step 10). The goal of this phase was to check the themes back to the original primary study data. The 50 primary studies were read again and coded according to the themes identified in order to ensure that each theme was taken into account in the analysis of each primary study. During this phase, the themes were consolidated and organized according to the classification schema described in Section 4.5. The themes then were synthesized (results presented in Sections 5 and 6). A theme owner was nominated for each theme, which was the researcher responsible for synthesizing the content of that theme. A second researcher reviewed the synthesis of the theme to improve its reliability. Definitions, benefits and challenges were identified and analysed in a similarly deductive manner.

3.7. Validity threats and limitations of the study

A strength of this study is that a number of researchers were closely involved, allowing for triangulation in all phases of the research. Seven researchers (the first seven authors) actively participated in the review, and three researchers with experience in conducting SLRs (the last three authors) acted as external reviewers to validate the research protocol and guide the research process. Nonetheless, there are some potential threats to the validity of this mapping study and these need to be considered when interpreting the results (Fig. 1, step 11b). We next describe these threats together with the strategies that were applied in order to mitigate their effects.

3.7.1. Identification of primary studies

The process of identifying the primary studies that constitute a mapping study is critical for the success of the research. The search string was built on three main attributes: deployment, continuity and speed in the context of the software industry (and their closely related terms). Nonetheless, the threat of missing relevant articles remains. The attempt to identify the entire body of knowledge in an emerging topic, such as CD, is very challenging. Inconsistency or the use of different terminology with respect to the search string (see Table 2) might have biased the identification of primary studies. This, however, is a minor threat because of the large volume of retrieved studies (21,382). In addition, the search string was used to search in keywords, titles and abstracts. We did not attempt to design a very precise search string that avoided noise because of the blurred nature of the phenomenon itself and because we were able to manage the number of retrieved studies. Thus, our strategy focused on retrieving as many documents as possible that were related to CD. For example, as explained in Section 3.2, we decided to separate the terms deployment and continuous, which created noise in the studies retrieved. From 15,062 publications (after the removal of duplicates and non-English documents), only 50 were selected as final primary studies. This low precision represents a moderate threat to the validity of the mapping study because it induced a significantly higher level of effort when selecting the final primary studies. However, seven researchers actively participated in the research, which minimizes the influence of this threat. In addition, three researchers individually piloted the inclusion and exclusion criteria in order to check their validity. Every step in the selection process was conducted by pairs of researchers in order to minimize subjectivity. When opinions within the pair conflicted, a third researcher helped to find agreement. In unclear cases, we were conservative and always included the paper in the next screening step.

3.7.2. Data extraction

When the primary studies were selected, they were subject to in-depth analysis. The analysis phase also posed threats to validity, which also need to be considered. These threats are largely because of researcher bias. The main countermeasures taken to address this threat were researcher triangulation and explicit definitions of the data to be extracted. Three aspects were analysed during the data extraction: study properties, quality assessment and factor analysis (including definitions of CD, benefits and challenges). Regarding the properties, each property was explicitly defined in the protocol, as indicated in Section 3.5. In addition, each paper was analysed by one researcher and then reviewed by a second researcher, who double-checked that the properties were properly collected. In cases of disagreement, a third researcher worked to achieve a resolution. The same process was applied in assessing both aspects of the study's quality: industrial relevance and scientific rigour. It is important to note that in the case of study quality, the evaluation depended on the reporting quality and not on the intrinsic quality of the study itself. With regard to the factor analysis, different countermeasures were used in the inductive coding, deductive coding and study synthesis. A single researcher inductively coded each primary study. This phase was deliberately unrestricted to produce *'free-codes'*. However, two full-day workshops consolidated the inductive coding to generate a reliable classification schema (themes identification), which involved seven researchers. Once themes were identified, they were formally defined for inclusion in the protocol by two researchers in order to ensure that all researchers involved had the same understanding of the themes. Based on the themes identified, the primary studies were deductively coded at the theme level. Deductive coding was conducted by a single researcher, who was nominated as the owner of that specific theme with a second researcher reviewing the theme level coding. When the primary studies were coded at the theme level, synthesis was carried out by the theme owner and then reviewed by two other researchers. In addition, one researcher, who had a global vision of the study, reviewed every step in the analysis in order to consolidate the results and ensure consistency of the analysis.

3.7.3. Publication bias

Publication bias refers to *'the problem that positive research outcomes are more likely to be published than negative ones'* (Unterkalmsteiner et al., 2012). This problem occurs in any literature review or mapping study. In our case, its effect was moderate because our study does not aim to compare research outcomes but to draw a map of the state of the art of CD. Nonetheless, publication bias may have affected our results regarding the benefits and challenges experienced when migrating to CD. The benefits may be overemphasized, compared to the possible risks. In addition, publication bias is affected by the sources of information considered in the study. However, we did not restrict publishers, journals or conferences. We also used five electronic databases, of which two were specialized in the field (ACM Digital Library and IEEE Xplore)

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

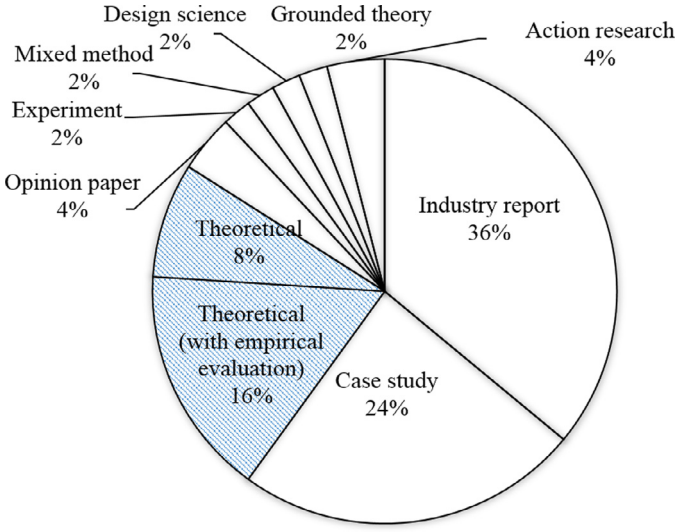


Fig. 3. Publication distribution-research method.

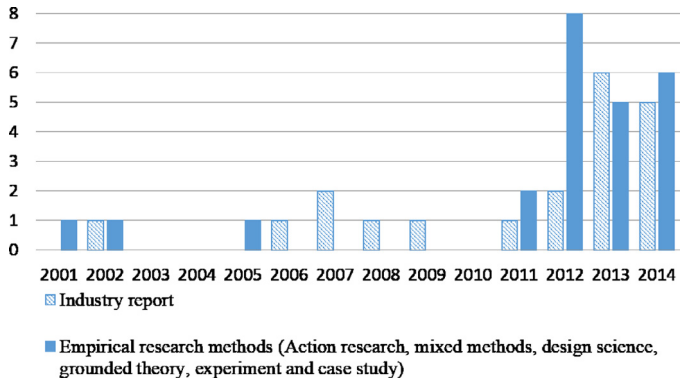


Fig. 4. Distribution of research method and publication year.

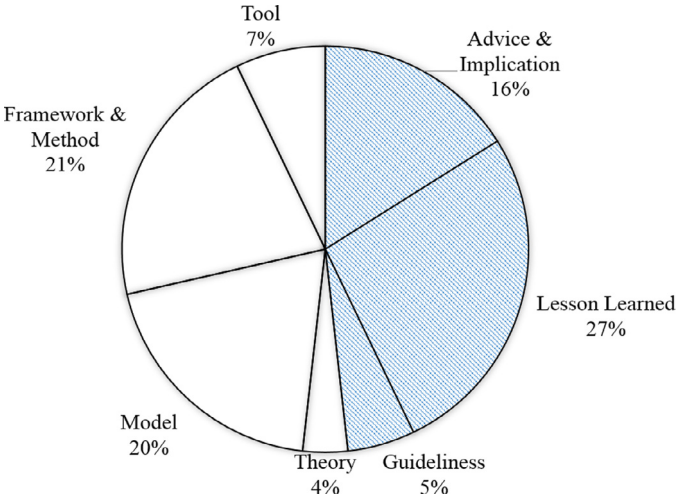


Fig. 5. Publication distribution-contribution.

and three offered wide coverage of diverse sciences (ISI Web of Science, Scopus and Science Direct). Although our results were limited by scientific studies published in these databases, they covered a wide range of the software engineering literature. In addition, we excluded non-peer reviewed scientific studies, book, book chapters and short papers because we did not consider that they would provide reliable information for our study.

4. Results: overview of the state-of-the-art of CD

From the initial set of 21,382 publications (see Table 3), 50 studies were identified as contributing to the topic of CD in the software domain. This section presents an overview of the body of CD knowledge found from their review. We structure this section according to the research questions presented in Table 1. Sections 4.1–4.4 present the research methods used, the kind of contributions provided, publications channels and the results of the quality assessment. Section 4.5 then presents a list of the main factors (i.e., recurrent themes in the literature) that characterize CD (classification schema). Finally, Section 4.6 presents an overview of the benefits and challenges of CD identified in the literature. The main factors, benefits and challenges are further elaborated in Sections 5 and 6, respectively. Table B.10 in Appendix B presents a detailed overview of each primary study.

4.1. RQ1.1: research methods

The primary studies were classified according to the research method used in the study, as defined in Appendix A. Fig. 3 shows the distribution of the research methods. Most studies on the state-of-the-art of CD were empirical in nature (72%, comprising case study, action research, grounded theory, design science, mixed methods, experiment as well as industry report.). However, theoretical studies (painted area in the chart), mainly in the form of models and frameworks, and methods, were also significant (24%, from which 16% were also empirically evaluated). Opinion papers completed the distribution of the research methods (4%). Interestingly, a high percentage of primary studies shared practitioners' experiences regarding the application of CD in the form of industry reports (36%). Even so, 48% of the studies used quite rigorous empirical research methods such as case studies, action research, grounded theory, design science, experiments and mixed methods. Case studies constituted a clear majority of this group (36%, of which 70% were pure empirical research that applied the case study research method, and 30% were combinations of theoretical research, mainly frameworks and models, evaluated by case studies). Action research (4%), mixed method, design science, grounded theory and experimentation (2% each) completed the list of empirical research methods. In general, the results showed that from a scientific point of view, the body of CD knowledge is still at an exploratory stage as a high percentage of the studies presented the views of practitioners regarding CD. This is natural in a phenomenon that has been especially driven by industry instead of resulting from the context of a research lab. However, as shown in Fig. 4, the percentage of studies that apply more rigorous research methods has been increasing in recent years.

4.2. RQ1.2: contributions

Fig. 5 shows the distribution of the research contributions. Many contributions were in the form of CD advice and implications (16%), lessons learned (27%) and guidelines (5%) when applying CD. Nonetheless, 48% of the contributions provided concrete approaches that could be used to support CD. These approaches included methods and frameworks for implementing CD (21%) (e.g. continuous Scrum (Agarwal, 2011) and methods for identifying risk areas in the context of CD (Comas et al., 2011; Antinyan et al., 2014)), models representing relevant concepts of CD (20%) (e.g., models for continuous experimentation (Fagerholm et al., 2014) and (Eklund and Bosch, 2012)), and tools supporting the technical infrastructure of CD (7%) (e.g. Gamma tool for assisting developers to monitor deployed systems (Orso et al., 2002)). The complete list of frameworks and methods, models and tools identified in the mapping study is listed in Table B.11, Appendix B.

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

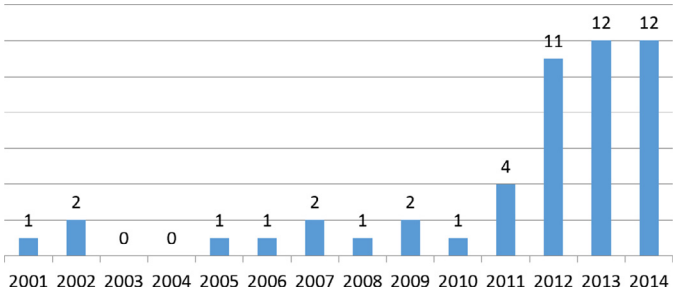


Fig. 6. Publication distribution by year.

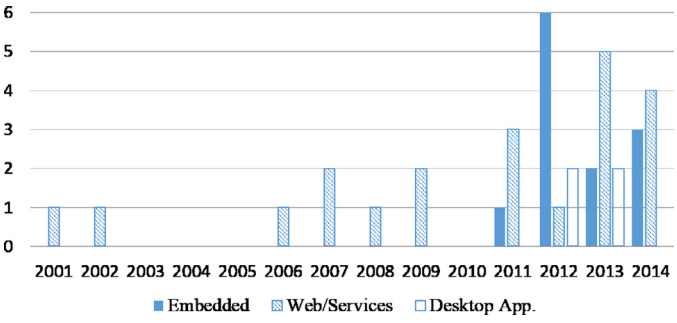


Fig. 7. Distribution by publication year and domain.

4.3. RQ1.3: publication years' frequency distribution and publication channels

As showed in Fig. 6, the papers reviewed were published between 2001 and 2014, which indicates the novelty of the phenomenon. Hence, the research on CD is still in its infancy when compared with the history of the software engineering discipline. Although some studies were published between 2001 and 2011, most were published within the last three years (68%). The publication of the book on CD by Humble and Farley (2010) has probably influenced the emergence of scientific studies on the topic. Still, the large number of recently published studies indicates an increasing interest in CD and points to the relevance of the area.

An interesting result was the distribution of the study domain by year (see Fig. 7) which was categorized in embedded systems, web/Internet based applications or services, and desktop applications. The category 'not stated' (N/S) was used for cases in which the domain was not clearly defined (see Section 3.5.1). As shown in Fig. 7, a clear majority of the primary studies were conducted in the web applications/services domain (42%). 24% of the studies were in the embedded systems and four studies (8%) were conducted in the context of desktop applications (i.e. Firefox). Finally, a high percentage of studies (26%) did not clearly describe the domain in which the research was conducted. Interestingly, the study domain and their frequency distribution by year indicated that the first studies published in the area focused only on web applications or services (primary studies published between 2001 and 2009). Thus, the first of our primary studies in the embedded domain was not published until 2011. Furthermore, the results of the pertinence of these studies (whether the study was devoted to CD or had a broader perspective, see Section 3.5.1), showed that 66% of studies conducted in the web domain fully focused on CD (14 studies), whereas this number decreased to 41% in the case of embedded systems (5 studies). These results indicate that CD is used more often in web-based applications. Some organizations in the area of web applications are currently able to deploy many new versions per day (see the references to organizational white papers and on-line blogs in Section 2.1). However, this goal is still a challenge for systems and services in other domains and organizations, such as in the field of embedded systems. Hence, primary studies in the embedded domain are more dedicated to specific aspects of CD. The challenges encountered in applying CD in embedded systems are further elaborated in Section 6.

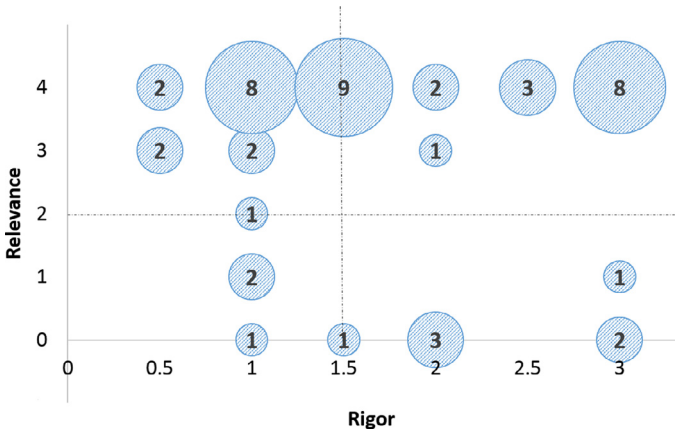


Fig. 8. Rigour-relevance overview.

Regarding publication channels, 84% of the primary studies were published in conference proceedings (42 papers), while 16% (8 papers) were published in journals or magazines. Overall, journal publications are subject to a more rigorous review process to ensure the quality of the research. However, they also take a longer time to be published and, this might discourage authors in a field that is developing rapidly and is largely driven by industry. Although the publication channels cannot be used to provide direct measure of the quality of the studies, it can be interpreted as an early indicator. The next section further elaborates on the scientific quality of the primary studies.

4.4. RQ1.4: primary study quality

An assessment of the primary studies' quality regarding scientific rigour and industrial relevance, as described in Section 3.5.2, resulted in Fig. 8. The two opinion papers (Poppendieck and Cusumano, 2012 and Benefield, 2009) were excluded from this analysis as they do not suit the assessment dimensions considered in Ivarsson and Gorschek's approach (Ivarsson and Gorschek, 2011). The raw data for this figure is available in Table B.10, Appendix B.

Regarding industry relevance, 37 primary studies (74%) had a relevance rating higher than two. Accordingly, most of the studies were conducted in industrial settings involving practitioners on an industrial scale. From these studies, 14 (28%) lie in the upper right quadrant of the chart (rigour ≥ 2, relevance ≥ 3). However, 23 studies (46%) exhibited high industry relevance (relevance ≥ 3) but showed low scientific rigour (rigour < 2). In addition, theoretical contributions without any empirical evaluation and research conducted with students constituted most of the studies with low relevance. Overall, it can be said that the topic of CD is highly relevant from the perspective of the industry; therefore, CD appears to be a promising research area because in an applied research field such as software engineering, it is the industry that ultimately determines the relevance of the research results.

Regarding scientific rigour, as many as 28 of the 50 primary studies have a rigour value of < 2 (56%). Consequently, the global scientific evidence of the body of CD knowledge can be considered as medium-low. With regard to context description, in the majority of the studies, the context in which the research is performed is not described to a degree such that it can be fully understood and compared with other contexts. Study design, data collection and data analysis were, in general, not well described, as many

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

Table 5
Continuous deployment factors – mapping study recurrent themes.

Factor	Description	Frequency	Primary studies
1. Fast and frequent release	Ability to release software whenever the organization wants to (on demand or at will) based on need and with preference given to shorter cycles or even continuous flow (weekly or daily).	28	(Bellomo et al., 2013b; Trimble and Webster, 2013, 2012; Van Der Storm, 2005; Nagy et al., 2010; Brown et al., 2013, 2011; Eklund and Bosch, 2012; Agarwal, 2011; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Goodman and Elbaz, 2008; Antinyan et al., 2014; Krusche and Alperowitz, 2014; Krusche et al., 2014; Lacoste, 2009; Ludwig et al., 2014; Marschall, 2007; Neely and Stolt, 2013; Olsson et al., 2013; Poppendieck and Cusumano, 2012; Cantor and Royce, 2014; Comas et al., 2011; Feitelson et al., 2013; Kalantar et al., 2014; MacCormack, 2001; Middleton and Joyce, 2012; Khomh et al., 2012; Lavoie and Merlo, 2013; Mantyla et al., 2013; Staron et al., 2012)
2. Flexible product design and architecture	CD requires evolutionary and robust software architecture with the aim of balancing speed and stability.	9	(Antinyan et al., 2014; Bellomo et al., 2013a, 2013b; Bosch and Eklund, 2012; MacCormack, 2001; Olsson et al., 2013; Van Der Storm, 2005; Brown et al., 2013; 2011)
3. Continuous testing and quality assurance	Ensuring the quality of the software at all times without compromises despite the need for fast and continuous deployment.	31	(Agarwal, 2011; Antinyan et al., 2014; Bellomo et al., 2013b; Blotner, 2002; Bosch and Eklund, 2012; Comas et al., 2011; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Goodman and Elbaz, 2008; Humble et al., 2006; Kalantar et al., 2014; Krusche et al., 2014; Lacoste, 2009; MacCormack, 2001; Marschall, 2007; Middleton and Joyce, 2012; Meyer et al., 2013; Neely and Stolt, 2013; Olsson et al., 2012; Trimble and Webster, 2012, 2013; Ard et al., 2014; Benefield, 2009; Brown et al., 2013, 2011; Khomh et al., 2012; Mantyla et al., 2013; Nilsson et al., 2014; Staron et al., 2012; Lavoie and Merlo, 2013; Baysal et al., 2012)
4. Automation	Automating the delivery pipeline from building and testing to deployment and monitoring.	24	(Agarwal, 2011; Antinyan et al., 2014; Blotner, 2002; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Goodman and Elbaz, 2008; Humble et al., 2006; Kalantar et al., 2014; Krusche et al., 2014; Lacoste, 2009; MacCormack, 2001; Marschall, 2007; Middleton and Joyce, 2012; Meyer et al., 2013; Neely and Stolt, 2013; Olsson et al., 2012, 2013; Trimble and Webster, 2013, 2012; Van Der Storm, 2005; Benefield, 2009; Poppendieck and Cusumano, 2012; Ard et al., 2014; Scacchi and Alspaugh, 2013)
5. Configuration management	Version control branching strategies and system configuration management approaches to enable CD.	12	(Goodman and Elbaz, 2008; Krusche and Alperowitz, 2014; Krusche et al., 2014; Marschall, 2007; Neely and Stolt, 2013; Feitelson et al., 2013; Meyer et al., 2013; Humble et al., 2006; Kalantar et al., 2014; Benefield, 2009; MacCormack, 2001; Van Der Storm, 2005)
6. Customer involvement	Mechanisms to involve customers in the development process and collect customer feedback from deliveries as early as possible (even near real-time) to drive design decisions and innovation.	12	(Van Der Storm, 2005; Eklund and Bosch, 2012; Bosch and Eklund, 2012; Feitelson et al., 2013; Ko et al., 2011; Krusche and Alperowitz, 2014; Krusche et al., 2014; MacCormack, 2001; Marschall, 2007; Olsson et al., 2013; Lavoie and Merlo, 2013; Zade and Choppella, 2012)
7. Continuous and rapid experimentation	Systematical design and execution of small field experiments to guide product development and accelerate innovation.	10	(Bosch and Eklund, 2012; Fagerholm et al., 2014; Feitelson et al., 2013; Neely and Stolt, 2013; Olsson et al., 2012, 2013; Benefield, 2009; Eklund and Bosch, 2012; Poppendieck and Cusumano, 2012; Goel et al., 2014)
8. Post-deployment activities	Activities that are conducted once the product (or a new feature or enhancement of the product) has been deployed to support fast business and technical decision making.	12	(Agarwal, 2011; Cukier, 2013; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Ko et al., 2011; Krusche and Alperowitz, 2014; MacCormack, 2001; Neely and Stolt, 2013; Olsson et al., 2012; Orso et al., 2002; Benefield, 2009; Goel et al., 2014)
9. Agile and lean	Extending agile and lean software development towards continuous flow to support CD.	22	(Bellomo et al., 2013a, 2013b; Trimble and Webster, 2013, 2012; Blotner, 2002; Brown et al., 2013; Agarwal, 2011; Fagerholm et al., 2014; Goodman and Elbaz, 2008; Gotel and Leip, 2007; Antinyan et al., 2014; Krusche and Alperowitz, 2014; Krusche et al., 2014; Marschall, 2007; Neely and Stolt, 2013; Olsson et al., 2013; Poppendieck and Cusumano, 2012; Ard et al., 2014; Cantor and Royce, 2014; Feitelson et al., 2013; Middleton and Joyce, 2012; Cantor and Royce, 2014)
10. Organizational factors	Organizational factors that enable CD (integrated corporative functions, transparency and innovative and experimental organizational culture).	17	(Bellomo et al., 2013a; Cukier, 2013; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Gotel and Leip, 2007; Kalantar et al., 2014; Krusche and Alperowitz, 2014; Krusche et al., 2014; Ludwig et al., 2014; Marschall, 2007; Middleton and Joyce, 2012; Neely and Stolt, 2013; Olsson et al., 2013; Brown et al., 2013; Poppendieck and Cusumano, 2012; Papatheocharous et al., 2014; Staron et al., 2012)

of the publications were industry reports. In addition, only twelve primary studies included a proper validity discussion (i.e. issues of bias, validity and reliability), whilst four primary studies mentioned validity, it was not described in detail. Thus, there is no description of threats to validity in 34 of the primary studies. Therefore, it can be concluded that there are important limitations to the scientific quality of the studies we retrieved and this inevitably reduces the reliability of the results.

4.5. RQ2: continuous deployment factors

The 50 primary studies cover a wide range of research topics. We categorized them into ten main themes, which together characterize the phenomenon of CD. A brief description of each theme and the primary studies that make reference to each theme are presented in Table 5. These factors are further elaborated in Section 5.

Table 6
Benefits of continuous deployment.

Benefit	PS
Shorter time-to-market	(Agarwal, 2011; Blotner, 2002; Feitelson et al., 2013; Goodman and Elbaz, 2008; Krusche and Alperowitz, 2014; Ludwig et al., 2014; Middleton and Joyce, 2012; Olsson et al., 2012, 2013; Neely and Stolt, 2013; Trimble and Webster, 2013, 2012; Benefield, 2009; Lacoste, 2009; Marschall, 2007; Neely and Stolt, 2013; Poppendieck and Cusumano, 2012; Fagerholm et al., 2014; Khomh et al., 2012; Lavoie and Merlo, 2013)
Continuous feedback	(Gotel and Leip, 2007; Krusche and Alperowitz, 2014; Krusche et al., 2014; Ludwig et al., 2014; MacCormack, 2001; Neely and Stolt, 2013; Trimble and Webster, 2012; Olsson et al., 2013, 2012; Cukier, 2013; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Benefield, 2009; Khomh et al., 2012)
Improved release reliability	(Agarwal, 2011; Humble et al., 2006; Krusche and Alperowitz, 2014; Neely and Stolt, 2013; Benefield, 2009)
Increased customer satisfaction	(Agarwal, 2011; Blotner, 2002; Neely and Stolt, 2013; Trimble and Webster, 2013; Khomh et al., 2012)
Improved developer productivity	(Agarwal, 2011; Gotel and Leip, 2007; Humble et al., 2006; Benefield, 2009)
Rapid innovation	(Feitelson et al., 2013; Goodman and Elbaz, 2008; Olsson et al., 2012, 2013)
Narrower test focus	(Comas et al., 2011; Feitelson et al., 2013; Neely and Stolt, 2013; Khomh et al., 2012)

Table 7
Challenges of continuous deployment.

Challenge	PS
Transforming towards CD	(Brown et al., 2013; Marschall, 2007; Blotner, 2002; Middleton and Joyce, 2012; Neely and Stolt, 2013; Olsson et al., 2012; Papatheocharous et al., 2014)
Customer unwillingness	(Olsson et al., 2013, 2012; Blotner, 2002; Agarwal, 2011; Orso et al., 2002; Zade and Choppella, 2012)
Increased QA effort	(Agarwal, 2011; Kalantar et al., 2014; Krusche and Alperowitz, 2014; Marschall, 2007; Meyer et al., 2013; Neely and Stolt, 2013; Khomh et al., 2012; Mantyla et al., 2013)
CD in embedded domain	(Trimble and Webster, 2012; Ard et al., 2014; Bosch and Eklund, 2012; Lavoie and Merlo, 2013)

4.6. RQ3: benefits and challenges

The benefits and challenges reported in the primary studies were identified and synthesized, as reported in Tables 6 and 7.

The benefits experienced when applying CD include shorter time-to-market, instant feedback, especially from customers when using proper monitoring and experimentation systems, improved release reliability, partially as a result of narrower test focus, and improved customer satisfaction and developer productivity. Overall, CD benefits are more often mentioned in the primary studies than are the challenges; this may be a consequence of authors and practitioners willingness to report positive rather than negative results. Still, important challenges were identified regarding the change that moving towards CD implies to the whole organization. These include customers' unwillingness to receive continuous product updates, increased QA efforts and difficulties applying CD in the embedded domain. CD benefits and challenges are further elaborated in Section 6.

5. Analysis of continuous deployment factors (RQ2)

First, we were interested in understanding what researchers mean when they refer to the term 'CD' (RQ2.1). Table 8 shows descriptions of continuous deployment and continuous delivery as they appear in the primary studies. We did not find any formal definition of CD in any of the primary studies; however, there appears to be some level of agreement amongst authors that CD refers to the ability of an organization to release software functionality directly to customers on demand and at will (deployment), faster and more frequently than traditional software development. We observed that there exists a tendency to use the two concepts interchangeably (except Fitzgerald and Stol, 2014).

In addition, the CD literature encompasses diverse recurrent themes or factors, as listed in Section 4.5. In the remainder of this section, we analyse each of the identified factors based on a synthesis of the primary studies.

5.1. Fast and frequent release

Several papers in the mapping study discuss fast and frequent release as shown in Table 5. Fast release is the ability to release software whenever the organization wants to, based on their need, which could be weekly or daily (Neely and Stolt, 2013; Krusche and Alperowitz, 2014). Almost all of our primary studies make reference in one way or another to accelerating the release cycle by shortening the release cadence and turning it into a continuous flow e.g. from months to weeks (Trimble and Webster, 2012; 2013; Khomh et al., 2012; Lavoie and Merlo, 2013), or from six months (Marschall, 2007) or eight-weeks (Neely and Stolt, 2013) to a continuous flow. However, achieving fast release in the form of a continuous flow is not free of charge. For example, Rally Software (Neely and Stolt, 2013) began to shrink the release cycles down to fortnightly, weekly, semi-weekly and finally at-will, which took months of preparatory work to streamline and automate the deployment process. In addition, a case study at Mozilla Firefox (Khomh et al., 2012) points out the question whether the quality of the software product improves as the shorter release cycles results in shorter testing periods. Also, Lavoie and Merlo (2013) claim that accelerating the release cycle can make it harder to perform re-engineering activities.

5.1.1. Continuous planning

The CD literature emphasizes two aspectsrelated to planning fast and frequent releases: continuity (e.g. Fitzgerald and Stol, 2014; Poppendieck and Cusumano, 2012) and taking a holistic view of planning (Fagerholm et al., 2014; Nagy et al., 2010; Brown et al., 2011; Bellomo et al., 2013b; Krusche et al., 2014). Traditional planning tends to be performed cyclically and is usually triggered by the annual financial year. However, CD challenges and changes traditional planning towards continuous planning in order to achieve fast and frequent releases (Fitzgerald and Stol, 2014). CD requires that planning activities are done more frequently to ensure alignment between the needs of the business context and software development, requiring tighter integration between planning and execution. Fitzgerald and Stol (2014) define plans as dynamic open-ended artefacts that evolve in response to changes in the business environment and require multiple stakeholders to be involved both from business and software functions. Hence, tighter integration between planning and execution is required in order to achieve a more holistic view of planning.

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

Table 8
Descriptions of continuous delivery and continuous deployment available in the primary studies.

PS	Description
Krusche and Alperowitz (2014)	'Continuous delivery is a set of practices and principles to release software faster and more frequently'.
Neely and Stolt (2013)	'Officially, we describe continuous delivery as the ability to release software whenever we want. This could be weekly or daily deployments to production; it could mean every check-in goes straight to production. The frequency is not our deciding factor. It is the ability to deploy at will'.
Olsson et al. (2012)	'The concept of continuous deployment, i.e. the ability to deliver software functionality frequently to customers and subsequently, the ability to continuously learn from real-time customer usage of software'.
Feitelson et al. (2013)	'. . . the practices that Internet companies use are known as continuous deployment. This reflects the habit of deploying new code as a series of small changes as soon as it is ready'.
Olsson et al. (2013)	'Continuous deployment is the idea that you push out changes to the code all the time instead of doing large builds and having planned releases of large chunks of functionality'.
Kalantar et al. (2014)	'. . . continuous delivery [1] that is, to continuously deploy the environment in a test environment that is reasonably similar to the actual production environment as part of development and testing efforts and to promote it to production when appropriate'.
Fitzgerald and Stol (2014)	'These concepts are related in that continuous deployment is a prerequisite for continuous delivery, but the reverse is not necessarily the case. That is, continuous delivery refers to releasing valid software builds to users automatically, whereas continuous deployment refers to the practice of deploying the software to some environment, but not automatically delivering to customers'.
Agarwal (2011)	'SaaS products provide an opportunity to provide consumers with continuous deployment of new features, as opposed to scheduled version upgrades as is the norm for products installed on-premise . . . continuous deployment of new versions of a software product in production'.

Poppendieck and Cusumano (2012) suggest considering software as a flow system where software is designed, developed, and delivered with a steady flow of small changes. A view that is fundamentally different from thinking of software development as a completed project, or even thinking about software as a series of annual or semi-annual releases. Rapid delivery should not be isolated to the software development alone, and the flow should happen within the overall product development cycle, of which software is just one aspect. Therefore, continuous planning includes all activities from strategic and business planning to product, portfolio and release planning. Similarly, Fagerholm et al. (2014) point out that according to software development based on continuous experimentation (common in CD, see Section 5.7), the experimental results should be continuously linked with the product roadmap as well as managed within a flexible business strategy in order to provide guidance for planning activities.

5.1.2. Mechanisms for achieving fast and frequent release

Besides continuous planning, other mechanisms are proposed in the literature to achieve fast and frequent release. Many of these mechanisms are important enough to become themes in their own right and are further developed in the following sections. For example, most studies highlight automation as essential to achieving fast and frequent release (e.g. Agarwal, 2011; Neely and Stolt, 2013; Marschall, 2007; Van Der Storm, 2005). Close interaction with customers (e.g. Trimble and Webster, 2012; Eklund and Bosch, 2012; Nagy et al., 2010), having a clear release process (e.g. Ludwig et al., 2014; Krusche et al., 2014), a release management workflow (Krusche et al., 2014) or a continuous delivery workflow (Krusche and Alperowitz, 2014; Krusche et al., 2014) appear also in the literature as enablers of CD. Staron et al. (2012) present a release readiness indicator, a mechanism to predict in which week the release would be possible, given the defect history. In addition, some studies discuss fast release in the context of ASD. For example, Krusche and Alperowitz (2014) distinguish between time-based and event-based delivery. They claim that fast release, as the teams' ability to deliver a potentially shippable product increment at any time in the project, is particularly useful at the end of the sprint (time-based delivery) when delivering the increment to the customer, but it also helps to obtain rapid feedback during the sprint (event-based delivery). In a similar vein, Agarwal (2011) proposes Continuous Scrum as a mechanism to achieve and sustain a rapid release cycle (one week product deployment) through parallel development.

5.1.3. Effects of fast and frequent release on product quality

The effects of fast and frequent release on the quality of the delivered software are also a focus in the literature. The CD literature highlights that a faster and more frequent release cycle should not compromise quality (e.g. Agarwal, 2011; Neely and Stolt, 2013; Feitelson et al., 2013; Khomh et al., 2012). Thus, it is important for the engineering and QA teams to ensure backward compatibility of enhancements, so that users perceive only improvements rather than experience any loss of functionality (Agarwal, 2011). Thus, the ability to release quickly does not mean that the development should be rushed into without a full understanding of what is actually being done (Neely and Stolt, 2013). Neely and Stolt (2013) advises monitoring everything to know the exact state of the system at every moment. Furthermore, based on a case study conducted on the effect of rapid releases upon quality at Firefox, Khomh et al. (2012) found that even though users do not experience significantly more post-release bugs in comparison with the traditional release model, program crashes occur earlier and users experience bugs earlier during execution. In another similar empirical study in the context of Mozilla Firefox, Lavoie and Merlo (2013) suggest that the rapid release model makes re-engineering activities harder to achieve and even though the code changes are smaller, they become a more important risk with a fast release cycle. In addition, the authors found that code change activities tend to focus more on bug fixing and maintenance than functionality expansion. However, there is no significant difference concerning the volume of changes among rapid release and a traditional model.

5.2. Flexible product design and architecture

Several primary studies make reference to product architecture and design (Antinyan et al., 2014; Bellomo et al., 2013a, 2013b; Bosch and Eklund, 2012; MacCormack, 2001; Olsson et al., 2013; Van Der Storm, 2005; Brown et al., 2013, 2011). CD demands a software architecture in which the product and its underlying infrastructure continuously evolve and adapt to changing requirements (Brown et al., 2013; MacCormack, 2001). Thus, it is essential that the underlying architecture is flexible and is able to accommodate rapid feedback (Bellomo et al., 2013a, 2013b; MacCormack, 2001). However, at the same time, the architecture must be robust enough to allow the organization to invest its resources in offensive initiatives (e.g. new functionality, product enhancements and innovation) rather than defensive efforts (e.g. bug fixes)

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

(Bellomo et al., 2013b, 2013a; MacCormack, 2001; Brown et al., 2013). To achieve this, the software architecture and design have to be highly modular and loosely coupled (MacCormack, 2001; Olsson et al., 2013; Bellomo et al., 2013b). In addition, what seems essential in the context of CD is that the software architecture accommodates mechanisms to rollback unsuccessful deployment (Olsson et al., 2013), supports independent deployment of a particular component rather than the entire system (Olsson et al., 2013; Van Der Storm, 2005) and enables experimentation through runtime variation of functionality as well as data collection mechanisms (Bosch and Eklund, 2012; Olsson et al., 2013).

The main challenge with regards to software design and architecture in the context of CD is the ability to maintain the right balance between speed (quickly delivering functionality to the users) and stability (providing reliable and flexible architecture) (Bellomo et al., 2013b; Brown et al., 2011). To overcome this challenge, some approaches propose a focus on measuring and monitoring source code and architectural quality. For instance, Bellomo et al. (2013a; 2013b) suggest extending prototyping to include quality attributes, such as performance or security-related issues (i.e. prototyping with a quality attribute focus), as a method of incorporating both functional and non-functional requirements in the context of CD. Rapid architecture trade-off analysis to accommodate rapid feedback and evaluate design options (Bellomo et al., 2013a), quantifying architectural dependencies by combining Design Structure Matrix (DSM) and Domain Mapping Matrix (DMM) techniques (Brown et al., 2011) and identifying and assessing risky areas of the source code based on diverse metrics (Antinyan et al., 2014) are also mechanisms proposed to maintain speed and stability. These mechanisms provide systematic feedback, bring more visibility and awareness to stakeholders and finally trigger re-factoring and re-architecture initiatives when needed (Bellomo et al., 2013b; Brown et al., 2011; Antinyan et al., 2014).

5.3. Continuous testing and quality assurance

Empirical evidence shows that most companies typically perform testing activities late in the development process causing unpredictable additional development efforts (Nilsson et al., 2014) and significant delays in releases (Staron et al., 2012). Many primary studies emphasize the importance of employing testing and quality assurance (QA) practices throughout the whole development process in the context of CD, as features are rolled out, and not just at the end of the development (e.g. Agarwal, 2011; Neely and Stolt, 2013; Blotner, 2002; Fitzgerald and Stol, 2014; Trimble and Webster, 2012; Goodman and Elbaz, 2008; Marschall, 2007). Thus, continuous testing aims to bring testing practices as close as possible to developers in order to avoid leaving testing activities only at the end of development (Fitzgerald and Stol, 2014).

In addition, a common problem in the testing activities is that many individual developers do not have an end-to-end overview of all the testing activities that are conducted during the software development process, other than their own individual activities (Nilsson et al., 2014; Baysal et al., 2012). As a consequence several problems such as duplication of test efforts and slow feedback loops are further testing challenges. To help alleviate these problems in the particular context of CD, authors emphasize the need to make all testing activities transparent to individual developers (Baysal et al., 2012) and using different techniques that help to describe and give a holistic overview of all testing activities such as CIViT – Continuous Integration Visualization Technique (Nilsson et al., 2014).

An important observation from studies conducted in open source software, particularly Firefox, was a slight increase on the number of reported bugs observed during testing in rapid release mode compared to traditional development (Khomh et al., 2012; Mantyla et al., 2013; Lavoie and Merlo, 2013). These studies on Firefox's transition to rapid releases also revealed that CD allows less time for testing activities but enables fast and thorough investigation of software features with the highest regressions risk at a relatively narrower scope (Khomh et al., 2012; Mantyla et al., 2013). The implication of the latter is that when transitioning to CD tremendous changes in terms of testing resources and strategies need to take place to make the testing process more sustainable in CD. Therefore, it remains crucial to assess whether such changes have significant impacts to the quality of the product (Mantyla et al., 2013).

On the other hand, improvements to existing testing practices in terms of fast feedback of code changes to developers during testing activities is expected in CD regardless of whether the software is open source or not (Baysal et al., 2012; Trimble and Webster, 2013). Reported strategies to continuously ensure the quality of the software in the context of CD include not only testing strategies but also creating a company culture of quality (Feitelson et al., 2013).

5.3.1. Test automation

Well-known testing techniques in ASD, such as test automation, are also crucial in CD in order to achieve continuous testing (Agarwal, 2011; Feitelson et al., 2013; Goodman and Elbaz, 2008; Marschall, 2007; Neely and Stolt, 2013; Olsson et al., 2012; Fitzgerald and Stol, 2014; Humble et al., 2006; Krusche et al., 2014; Meyer et al., 2013; Lacoste, 2009; Trimble and Webster, 2013; Benefield, 2009; Nilsson et al., 2014). According to our primary studies, test automation ensures: (a) quality of software through extensive test coverage (Goodman and Elbaz, 2008; Marschall, 2007), (b) continuous integration and release of quality software (Neely and Stolt, 2013; Humble et al., 2006; Krusche et al., 2014; Lacoste, 2009) and (c) provide early feedback to the development team (Humble et al., 2006) so that issues are resolved and root causes eliminated (Fitzgerald and Stol, 2014). In test automation, a variety of test suites: unit, functional, integration, and performance tests are executed at different phases (Humble et al., 2006) and with different scope: component, subsystem, partial product, product, release and customer (Nilsson et al., 2014). These test suites use practices such as automated execution of test scripts on a build server after each code commit to test and check the status of the code or the build (Agarwal, 2011; Lacoste, 2009; Marschall, 2007; Trimble and Webster, 2013), automating acceptance testing (Humble et al., 2006; Middleton and Joyce, 2012), automatic testing of the production environment in non-embedded software (Kalantar et al., 2014) and simulation to demonstrate and test the quality of software much earlier for embedded software (Ard et al., 2014). In addition, efficient prioritization and the ordering of automated test execution are emphasized in order to ensure fast feedback to the development team as well as the proper use of resources (Fitzgerald and Stol, 2014; Humble et al., 2006).

Two primary approaches that facilitate test automation, code driven testing (e.g. TDD and test planning (Neely and Stolt, 2013)) and GUI testing are discussed. Code driven testing practices are mostly preferred by practitioners working in CD (Agarwal, 2011; Bellomo et al., 2013b; Middleton and Joyce, 2012; Blotner, 2002; Neely and Stolt, 2013; Brown et al., 2013). However, due to limitations such as high costs and the effort required to train and manage test-programs, Agarwal (Agarwal, 2011) recommends the use of automated GUI testing. GUI testing is seen as a lead indicator of bugs that typically appear in a production environment (Neely and Stolt, 2013). More details on how GUI testing is implemented in practice are given in the Automation theme (Section 5.4).

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

5.3.2. Testing with users

Testing new features while in real use and using a small fraction of actual end users, is also emphasized when aiming to CD (Feitelson et al., 2013; Lacoste, 2009; Neely and Stolt, 2013; Bosch and Eklund, 2012; MacCormack, 2001; Khomh et al., 2012). This approach provides immediate feedback about the feature quality as perceived by users, allowing developers to quickly discover new bugs to fix (Lacoste, 2009; MacCormack, 2001; Khomh et al., 2012). For example, Facebook reports testing new features internally by employees and later by a subset of real world users (Feitelson et al., 2013), before making them available to all users. Similarly, in Lacoste (2009) testing new features in real use is done with beta users who are also actual users. When the result of testing features in real use is satisfactory, features are deployed to the entire user base. Otherwise, alternative mechanisms such as rollback are executed (Feitelson et al., 2013; Neely and Stolt, 2013). Furthermore, although this approach is mainly used in web applications (Feitelson et al., 2013; Lacoste, 2009), there is also evidence of testing in real use in the embedded domain, but with the aim of establishing a proof-of-concept as the system is not intended for mass production (Bosch and Eklund, 2012). See Customer involvement (Section 5.6) and Post-deployment activities (Section 5.8) themes for more details.

5.3.3. Creating a culture of quality

It is also suggested that continuous testing and QA embody a culture of developer responsibility in which developers bear the responsibility for writing good code, perform thorough tests as well as support the operational use of their software (Feitelson et al., 2013; Marschall, 2007; Trimble and Webster, 2013). As systems become larger and more complex, such culture complements test automation systems and allows the quality of the software to be maintained at scale.

5.3.4. Technical debt

Finally, another important aspect observed and related to quality assurance in the context of CD is the concept of technical debt. It was noted that as a consequence of trade-offs between the fast deployment of software and poor development, testing and quality assurance practices, organizations acquire technical debt over time (Middleton and Joyce, 2012; Bellomo et al., 2013b; Brown et al., 2011). As technical debt causes architecture quality degradation over time, organizations applying CD need to continuously monitor and measure the quality of the degrading architecture (Bellomo et al., 2013b; Brown et al., 2011). Interestingly, two authors in our review have separately studied measurement techniques to determine risky files in embedded software development (Antinyan et al., 2014) and the risk associated with deploying certain features (Comas et al., 2011). The method proposed by Antinyan et al. (2014) is based on measuring a set of code properties, such as McCabes cyclomatic complexity, in order to identify areas of source code that may be faulty and difficult to maintain in order to provide quick feedback to developers. Risk assessment scoring heuristics for software deployment is another method proposed by Comas et al. (2011). The method proposes decomposing web application into different functionality tiers to identify changes and the requirements needed to implement the changes. Each change implemented in the software is analysed along with risk. The result of the analysis is used to provide information to system integrators about the risky areas on which to focus their efforts e.g. testing in appropriate places.

5.4. Automation

In the context of CD, the focus is on automating the entire delivery pipeline. Prior to CD only parts of the pipeline were automated. However, the CD literature highlights the importance of eliminating all manual steps from build to deploy (Goodman and Elbaz, 2008; Neely and Stolt, 2013)extending continuous integration with release and deploy automation, and automated configuration management of deployment environments. Humble et al. (2006) recommend automating build, testing, and deployment in the early stage of the project and to evolve the automation along with the application. Furthermore, automation is also needed for measuring and improving the work, which concerns the whole delivery pipeline.

5.4.1. Continuous integration

Continuous integration (CI) is one of the main enablers of CD (Olsson et al., 2012, 2013; Goodman and Elbaz, 2008; Fitzgerald and Stol, 2014; Trimble and Webster, 2012; Ard et al., 2014; Agarwal, 2011; Humble et al., 2006; Krusche et al., 2014; Lacoste, 2009; Marschall, 2007). The main advantage of CI is that it automates tasks such as compiling code, running unit and acceptance tests, monitoring and validating code coverage, checking compliance with coding standards, static code analysis, automatic code review and building deployment packages (Fitzgerald and Stol, 2014; Agarwal, 2011; Marschall, 2007; Trimble and Webster, 2013; Ard et al., 2014). Therefore, CI provides mechanisms to ensure that there is always a shippable product that has passed all of the testing phases (Fitzgerald and Stol, 2014; Olsson et al., 2013). Fitzgerald and Stol (2014) report that the frequency of integration is as important as automation itself. Thus, the frequency should be high enough to ensure quick feedback to developers. CI is usually coupled with feedback mechanisms (e.g. dashboards) (Goodman and Elbaz, 2008; Humble et al., 2006; Lacoste, 2009; Marschall, 2007) that enable rapid feedback on source code and triggers for immediate problem resolutions (Fitzgerald and Stol, 2014; MacCormack, 2001). Regarding the application domain, the main challenge of using CD in embedded systems is that physical assets and hardware equipment should also support CI in order to benefit from CD as a whole. To overcome this problem, Ard et al. (2014), in the avionics domain, propose a simulated integrated system that facilitates CI of embedded systems.

The level of automation of CI system varies depending on the primary study (Trimble and Webster, 2013, 2012; Blotner, 2002; Agarwal, 2011; Goodman and Elbaz, 2008; Humble et al., 2006; Lacoste, 2009; Marschall, 2007; Neely and Stolt, 2013; Feitelson et al., 2013; Middleton and Joyce, 2012; Benefield, 2009; Poppendieck and Cusumano, 2012). Guidance and working practices on structuring the CI pipeline and what possible stages and tools can be used in the pipeline are reported in several primary studies (Goodman and Elbaz, 2008; Humble et al., 2006; Feitelson et al., 2013; Trimble and Webster, 2013, 2012; Lacoste, 2009; Agarwal, 2011; Antinyan et al., 2014). Goodman and Elbaz (2008) suggest automating quick builds in order to provide developers with instant feedback. Trimble and Webster (2013); Goodman and Elbaz (2008) perform nightly builds. In Trimble and Webster (2012); Lacoste (2009), the CI-system automatically closes the source control repository from further commits in case of a build failure. Humble et al. (2006) suggest using several testing stages in which different types of testing should be independent from each other. Facebook (Feitelson et al., 2013) integrates a code review stage as part of the build process using the tool Phabricator. Antinyan et al. (2014) develop an automatic measurement system to identify risky files that may need refactoring.

Automated GUI-testing seems to be a popular practice in our primary studies (Agarwal, 2011; Trimble and Webster, 2012, 2013; Feitelson et al., 2013; Neely and Stolt, 2013). In GUI testing *'test-automation software is used to record mouse movements and key-presses, and replay these when needed'* (Agarwal, 2011). However, implementing automated GUI testing in practice is challenging

(Agarwal, 2011) because it needs to be flexible enough to test different scenarios (Trimble and Webster, 2013) and do so at a fast-pace (Neely and Stolt, 2013). Agarwal (2011) also notes the need for manual observation during automatic GUI testing because the testing software is not able to identify the occurrence of (unexpected) errors due to difficulties in identifying errors automatically. Watir and Webdriver are used in Facebook for automated GUI testing (Feitelson et al., 2013).

5.4.2. Release and deploy automation

Many of our primary studies extend beyond traditional CI into release and deploy automation (Agarwal, 2011; Goodman and Elbaz, 2008; Humble et al., 2006; Neely and Stolt, 2013; Feitelson et al., 2013; Van Der Storm, 2005). As an example Agarwal (2011) suggest an automation system to integrate the configuration management, build, release and testing processes. The system provides developers with a means to migrate changes from one environment to another (e.g. development to production). Thus, the system would allow a release manager to perform a release with selected content, and to upgrade the software in the target environment. In a similar vein, Facebook (Feitelson et al., 2013) has a tool for deployment (Gatekeeper) that allows the developers to turn features on and off in the code, and to select which user groups see which features.

Various type of guidance, working practices, and tools for automated deployment and monitoring are provided in Humble et al. (2006); Kalantar et al. (2014); Meyer et al. (2013); Feitelson et al. (2013); Neely and Stolt (2013); Scacchi and Alspaugh (2013). Humble et al. (2006); Kalantar et al. (2014); Meyer et al. (2013) support automated deployment to all types of environments (development, test, staging, production). Feitelson et al. (2013) report on how Facebook performs multi-stage deployment where internal testing and performance testing are performed before deployment to production. Neely and Stolt (2013) report the experiences of Rally Software when mimicking test environments to production environment. This lowers the risk in deployment and provides advice to identify barriers that prevent delivery from a commit. DevOps also provided practices for automatically linking operations with development and QA functions. For example, with DevOps configuration management becomes another form of source code that can be automatically managed using standard source development techniques (Meyer et al., 2013). In addition, deployment monitoring tools for global deployment are discussed in Neely and Stolt (2013); Feitelson et al. (2013) (e.g. Feitelson et al., 2013 use BitTorrent). Although most of the primary studies acknowledge the benefits that automation brings in the context of CD, there were also studies that noted the possible limitations that automation might imply in terms of flexibility for adapting and configuring systems to particular organizations or development contexts (Scacchi and Alspaugh, 2013).

5.4.3. Automation of configuration management for deployment environments

The configuration management system (CM) is also automated in CD, which enables automated provisioning and deployment to various target environments. Meyer et al. (2013) describe an automated CM tool that specifies configuration actions using a high-level declarative language stored on a central server. Using this tool, client machines compare their configuration state to the central configuration specifications after which configuration actions are applied as necessary to bridge the gap between the current configuration and the desired configuration. Kalantar et al. (2014); Humble et al. (2006); Benefield (2009) also describe automated CM systems. For example, Kalantar et al. (2014) developed Weaver, a system to manage the configuration of an entire environment, including software components that span systems and the infrastructure elements that are needed to support them. To do this, they defined a Domain Specific Language to specify blueprint environments, and at runtime execute the blueprints to create or modify environments. The system allows validation of blueprints at design, deployment, and runtime. However, it is not intended to replace low level automation building blocks to install and configure individual software components (e.g. scripting languages, Chef, Puppet). Benefield (2009) suggests a similar system to Kalantar et al. (2014) in which they can manage configuration and deployment to a system. In addition, Humble et al. (2006); Kalantar et al. (2014); Meyer et al. (2013) treat deployment scripts (or blueprints) as code which is stored under version control and can also be subject to automated tests as noted in Meyer et al. (2013).

5.5. Configuration management

Besides automating CM, two main topics emerge in the review regarding CM and CD: version control branching strategies and system configuration management.

5.5.1. Version control branching strategies

Version control branching strategies aim for regular and incremental delivery, lower integration risk (due to more frequent merges) and improved coordination between teams. Goodman and Elbaz (2008); Krusche and Alperowitz (2014); Krusche et al. (2014); Marschall (2007); Neely and Stolt (2013); Feitelson et al. (2013) discuss version control branching strategies that are suitable for CD. Using a single branch, without any private branches, in order to keep the continuous deployment of new functionality simple is reported in Goodman and Elbaz (2008); Neely and Stolt (2013). However, most studies in the review report as using both main (or master) and separate branches. For example, Marschall (2007); Feitelson et al. (2013) report the use of a main branch that is kept releasable all the time. In addition, separate branches are used for each user story, which are merged into the main branch after passing quality gates. In a similar vein, Krusche and Alperowitz (2014); Krusche et al. (2014) report experiences with a branching model called git-flow.[15] In their variation of git-flow, features are developed in separate feature branches, after which they are merged into the development branch to be shared with other developers. Internal releases are triggered when developers merge feature branches into the main development branch (under the release manager's supervision). Neely and Stolt (2013) describe the use of a master branch, which encourages small size stories in order to make merges easier. Thus, long running branches or feature branches are rarely employed in this case. However, instead of feature branches the organization uses feature toggle through an administration interface to switch the toggles and clean them after a story is completed.

5.5.2. System configuration

Meyer et al. (2013); Humble et al. (2006); Kalantar et al. (2014) base CM on automation as described in Section 5.4.3. In addition, different strategies are proposed in the literature to manage system configuration in CD.

In order to make system CM easier, Humble et al. (2006) suggest deploying the same software binaries in every environment and maintaining runtime configuration separately from binaries. To facilitate the identification of problems and solutions, frequent deployments, where each deployment introduces only a limited amount of new code, characterize the software development process at Facebook (Feitelson et al., 2013). In case of problems they roll back single commits and any of their dependencies, or if that

[15] http://nvie.com/posts/a-successful-git-branching-model/ .

is not possible, they revert the whole binary (consisting of possibly multiple commits) to the previous working version (Feitelson et al., 2013). They also suggest reverting back commits of developers who are not present during the delivery in order to minimize deployment problems. In a similar vein, MacCormack (2001) highlights the importance of being able to trace feedback from a release to a particular revision of software. Benefield (2009) suggests an atomic packaging scheme where versioned self-contained packages can be independently released and rolled back. Finally, Van Der Storm (2005) develops a theoretical model for upgrading component-based software in which every release references all of its dependencies, and where releases can be tracked back to source code.

5.6. Customer involvement

One characteristic of CD is collecting customer feedback from deliveries as early as possible (even near real-time) in order to base design decisions on real customer usage and thus use customer input as the main driver for innovation (Olsson et al., 2013; Eklund and Bosch, 2012). The importance of customer feedback is highlighted not only in requirements elicitation, prioritization, definition of user stories and 'definition of done' (DoD) (Krusche et al., 2014; Bosch and Eklund, 2012; Feitelson et al., 2013) but also in other phases of the product development such as acceptance testing. For example, Marschall (2007) suggests developing customer tests in a way that the customer is required to sign off on each user story (or product feature) before it can be considered complete.

Customer involvement in CD concerns the following tasks: (1) determining from whom feedback is collected, (2) what issues feedback addresses, (3) how feedback is collected and in which format, (4) how feedback is processed and (5) how feedback is taken into account in the development process. Some approaches can be found in the CD literature regarding the three first items; however, approaches for processing feedback and taking it into account are scarce.

Regarding the first task, determining from whom feedback is collected, it depends on the kind of feedback that is required or interesting to gather. Olsson et al. (2013) propose locating lead customers who serve as role models for other customers. MacCormack (2001) suggests that a valuable avenue for identifying lead (beta) customers is through exploring the company's customer-support database. Ko et al. (2011) also warn about the importance of selecting a sample of customers that is representative of the user community and discusses challenges when using a vocal minority of existing users.

The second and third tasks refer to how to get customer feedback and limit it so that it targets only the specific issue at hand. Regarding getting feedback for bug fixes, Van Der Storm (2005) introduces a component-based system in which specific components are automatically delivered after fixing the bug and customer feedback is accurately collected. This allows for getting fast feedback for a specific issue on which a developer is focused at the time (Krusche and Alperowitz, 2014; Krusche et al., 2014). In Facebook, Gatekeeper is used to control which parts of the code are actually active for customers (Feitelson et al., 2013). With Gatekeeper, engineers can turn tests on and off at will and also apply them to selected user groups. In this way, the feedback is collected only from the active parts of the code. Gatekeeper can also be used to turn off new code that is causing problems, thereby reducing the need to immediately deploy a correction (Feitelson et al., 2013). Moreover, automated tools such as a delivery server or deployment pipeline allow customers to give feedback directly within the software in a structured way (Krusche and Alperowitz, 2014; Krusche et al., 2014). In addition, monitoring customer usage scenarios (even without the user knowing about it) (Bosch and Eklund, 2012); A/B testing as an experimental approach to find out what users want (Feitelson et al., 2013); and prototypes and mock-ups as the first visualization of user interface (MacCormack, 2001) are useful for collecting feedback and helping with an accurate understanding of customer expectations.

However, the literature does not provide significant solutions for tasks four and five, i.e., how the feedback is processed and how the feedback is taken into account in the development process. When collecting customer feedback, especially when using structured feedback channels, there needs to be mechanisms in place to process incoming feedback and to interpret the information quickly. Excepting MacCormack (2001), which proposes a procedure where the first thing that developers have to do in the morning is to check if there are problems in their latest submission regarding feedback for daily builds, mechanisms for systematically processing feedback were not elaborated in the primary studies. Nevertheless, the literature does emphasize close collaboration with the customer, especially during requirements elicitation, prioritization and the definition of user stories and DoD (Krusche et al., 2014; Bosch and Eklund, 2012). Moreover, there are warnings about the effects that continuous changes in a product might have upon customers. For example, Lavoie and Merlo (2013) notes that when doing fast releases, how much and what to change between releases must be seriously considered, as this has a direct impact on the interactions between users and developers. In the same vein, Zade and Choppella (2012) highlight that in fast releases, changes in user interfaces need to be provided with serious attention. If the way a user interacts with the software changes considerably between releases, then a negative impact on customer experience is likely.

5.7. Continuous and rapid experimentation

Ten primary studies, published during recent years, make reference to continuous and rapid experimentation (Feitelson et al., 2013; Eklund and Bosch, 2012; Bosch and Eklund, 2012; Poppendieck and Cusumano, 2012; Olsson et al., 2012, 2013; Benefield, 2009; Fagerholm et al., 2014; Neely and Stolt, 2013; Goel et al., 2014). Although the literature lacks a unified definition, continuous and rapid experimentation in the context of CD refers to systematically designing and executing small field experiments to guide product development and accelerate innovation; thus, it aims to base the business and design decisions of product enhancements and new functionality on data rather than on stakeholder opinions, even if they are experts in the area (Eklund and Bosch, 2012).

The 'Stairway to Heaven' model suggested R&D as an experimental system as the last step of its evolutionary path of software organizations from traditional methods to CD and beyond (Olsson et al., 2013, 2012). It describes a situation where the organization constantly conducts experiments to guide product development and accelerate innovation and decision-making. To achieve this objective, companies need to adopt a short-cycle innovation process centred on customer feedback and usage data (Olsson et al., 2012, 2013). This information is used to guide the evolution of the system and the actual deployment of new software functionality (Olsson et al., 2012, 2013). For example, Facebook uses A/B (split) testing as an experimental approach to immediately identify user needs and values rather than trying to elicit requirements following the traditional requirements engineering approach (Feitelson et al., 2013). When using A/B testing, randomized experiments are conducted over two or more variants of an enhancement (or similar feature) in order to compare how they are perceived by end-users through statistical hypothesis testing (Feitelson et al., 2013; Benefield, 2009). This experimental approach is largely facilitated by the fact that CD significantly reduces the gap between

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

the company and its customers (see Customer involvement theme, Section 5.6) (Benefield, 2009; Feitelson et al., 2013).

In order to enable continuous and rapid experimentation, architectural infrastructure for runtime variability of functionality, mechanisms for data collection and rollback mechanisms to revert changes are required (Olsson et al., 2013). For instance, Rally Software (Neely and Stolt, 2013) suggested A/B testing with Feature Toggle as a technique to manage and support run-time variability of functionality. Goel et al. (2014) describe the development of an infrastructure for fast upgrade of database systems which enabled Facebook to deploy experimental software builds and improvements on a large scale of machines without degrading the systems uptime and availability. Apart from technological requirements, organizational functions, which includes release and product management as well as innovation and R&D, must be well aligned and tightly integrated (Olsson et al., 2013, 2012).

It is interesting to observe that continuous experimentation has been proposed not only in the context of web applications in companies such as Facebook or Rally Software, where innovation cycles are naturally shorter, but also in the context of embedded systems. For example, Eklund and Bosch (2012) and Bosch and Eklund (2012), in the automotive industry, present the innovation experiment system (IES). IES is an evolution of current R&D practices moving from considering innovation as a process internally guided and assessed by the original system manufacturer to a process in which innovation is actually evaluated by real users at scale. However, the literature also recognizes the limitations of applying the experimental paradigm in the context of safety critical or other systems that require certification and heavy verification and validation processes (Bosch and Eklund, 2012; Fagerholm et al., 2014).

Regarding how continuous and rapid experiments are actually conducted, the CD literature is quite scarce. Fagerholm et al. (2014) present an initial model for continuous experimentation composed of an experimentation cycle based on build-measure-learn blocks and their underlying infrastructure. The build-measure-learn blocks structure the activity of conducting experiments and connect product vision, business strategy and technological product development through experimentation. On the other hand, the underlying infrastructure of the model comprises three layers, including roles involved in running the experiment, enabling technical infrastructure and information artefacts that are needed for conducting the experiments.

5.8. Post-deployment activities

The theme of post-deployment activities refers to those activities that are conducted once the product (or a new feature or enhancement of the product) has been deployed (Olsson et al., 2012). CD has created a large number of new opportunities not just for observing user behaviours and monitoring how systems and services are being used (Cukier, 2013; Benefield, 2009), but also for identifying unexpected patterns and runtime issues (Cukier, 2013; Krusche and Alperowitz, 2014), monitoring system quality attributes (Cukier, 2013; Feitelson et al., 2013; Fitzgerald and Stol, 2014) and collecting real-time data to feed both business and technical planning (Benefield, 2009). For instance, Orso et al. (2002) proposed an approach to perform different monitoring tasks and collect useful information on their software's behaviour. In addition, continuous monitoring might also aim to monitor metrics related to service-level agreements and system quality attributes, including performance, system availability and operational infrastructure. Goel et al. (2014) report on a very fast, distributed, in-memory database at the heart of Facebook that is extensively used for post deployment activities including advertisement revenue monitoring, performance debugging as well as real-time analysis of user behaviour and service logs.

As acknowledged by various studies (Cukier, 2013; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Benefield, 2009), the main objective of continuous monitoring is to constantly monitor and measure both business indicators and infrastructure-related metrics in order to facilitate and improve business and technical decision-making. More importantly, continuous monitoring must always be unobtrusive to users, thought it needs to be visible and accessible to all relevant stakeholders, including development, operation and business people (Cukier, 2013; Benefield, 2009). One of the most significant activities in this sense is post-release testing to ensure successful deployment (Agarwal, 2011) and also performing critical validation and testing on real users at scale (Feitelson et al., 2013). Dark deployment is used where enterprises deliver new features or services that are invisible to customers and have no impact on the running system. This technique can be used to test system quality attributes and examine them under simulated workload in a real production environment (Feitelson et al., 2013; Neely and Stolt, 2013). Another relevant practice, canary deployment, allows enterprises to deliver a new version to a limited user population to test the system under real production traffic and use. The new version is then delivered to the whole user population once it reaches a high enough level of quality (Feitelson et al., 2013; Neely and Stolt, 2013).

5.9. Agile and lean software development

CD goes beyond agile and lean software development; thus, agile and lean software development methods and practices are the first steps the organization can take toward CD, e.g. (Fitzgerald and Stol, 2014; Staron et al., 2012). Hence, ASD methods and practices can be considered as an enabler for CD. However, CD scales ASD practices throughout the whole organization instead of focusing only on team-level activities. For example, this is noticed in the continuous experimentation approach, in which software is developed based on field experiments with relevant stakeholders, i.e. customers or users (Fagerholm et al., 2014). The experimental results are linked throughout the organization with a product roadmap as well as managed within a flexible business strategy (Fagerholm et al., 2014). Along the same lines, lean software development promotes the consideration of the whole organization as part of CD: *'If you deliver daily, waste is exposed almost immediately . . . optimizing just a part of the system simply is not an option with daily deployment'* (Poppendieck and Cusumano, 2012). To accelerate continuous software delivery and achieve agility at scale, Cantor and Royce (2014) describe the IBM transformation from conventional engineering governance to economic governance and Bayesian analytics, which integrate governance with agility aspects.

Most of the primary studies discuss in one way or another aspects related to ASD. For example, Benefield (2009); Poppendieck and Cusumano (2012); Middleton and Joyce (2012) explain that lean software development supports delivery of a continuous flow of small features into production, as is the aim of CD. Benefield (2009) focuses on lean techniques for the SaaS delivery model. Using specific Agile and Lean software development methods appears frequently in the primary studies as well (e.g. using Scrum Bellomo et al., 2013a, continuous Scrum Agarwal, 2011, pair programming Blotner, 2002, eXtreme Programming Gotel and Leip, 2007, or Rugby which is an agile process model including workflows for the continuous delivery of software Krusche et al., 2014).

CD also changes the traditional ASD practices and methods into a continuous flow. Continuous ways of working are described in the ASD literature as follows: continuous improvement and employee empowerment, e.g. (Middleton and Joyce, 2012); CI, e.g. (Bellomo et al., 2013b) and (Krusche et al., 2014); continuous deliv-

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

ery, e.g. (Krusche et al., 2014; Krusche and Alperowitz, 2014; Neely and Stolt, 2013); continuous delivery of features, e.g. (Bellomo et al., 2013a); or transforming release cycles into a continuous flow, e.g. (Marschall, 2007) and (Agarwal, 2011). For instance, Agarwal (2011) reports results from continuous scrum in which bug fixes, minor enhancements and major features are released continuously on a weekly basis by a single development team. Each sprint has three phases, creating a triple-sprint overlap pattern: planning, development and QA. The development team is divided into team members who are responsible and capable of executing each of these phases; hence, there are sub-teams each with a different function, i.e. planning, development and QA. Each sprint is time-boxed into a three-week period. During the first week of a sprint, the product owner and scrum master together with the inputs from the development team formulate a plan for the remainder of the sprint. Thereafter, the planning sub-team starts planning the next sprints and the development team starts development on the sprint that was planned by the sub-team in the prior week. Similarly, the QA team performs QA on what was developed by the development team in the prior week. Thus, sequential sprints overlap with a phase-lag of a one-week release cycle (Agarwal, 2011).

5.10. Organizational factors

Many factors related to organizational aspects were also highlighted in the CD literature. We classified these aspects into three groups: integrated corporate functions, transparency and innovative and experimental organizational culture.

5.10.1. Integrated corporate functions

Both empirical and non-empirical studies stressed the integration of the R&D organization with other corporate functions such as sales, marketing, product management, QA, release and operations. This integration is crucial for fast delivery and deployment. It enables transparency and understanding of the whole picture of product development activities and overcome corporate constraints that often cause delays in product deliveries, e.g. hand-over delays and communication gaps (Gotel and Leip, 2007; Bellomo et al., 2013a; Ludwig et al., 2014; Poppendieck and Cusumano, 2012; Kalantar et al., 2014; Feitelson et al., 2013; Fitzgerald and Stol, 2014). A flat R&D organizational structure is common when applying ASD (Gotel and Leip, 2007; Neely and Stolt, 2013). However, CD demands a greater alignment of the R&D organization with other corporate functions (Olsson et al., 2013). For instance, integration of R&D with the operations/maintenance team, also referred to as DevOps, is noted in several studies (Feitelson et al., 2013; Brown et al., 2013; Fitzgerald and Stol, 2014; Krusche and Alperowitz, 2014). Similar to DevOps is the emphasis on the integration between software development and business strategy, termed BizDev (Fitzgerald and Stol, 2014). Both DevOps and BizDev focus on achieving a shorter cycle time with increased feedback loops (Cukier, 2013; Fitzgerald and Stol, 2014).

Several strategies describing how to integrate corporate functions are depicted in the CD literature; however, they are mostly initial proposals. For instance, Neely and Stolt (2013) describe the journey that Rally Software followed for tracking the status of work in real time to sales and marketing teams in order to integrate them with R&D. One challenge here is that in addition, the marketing strategy needs to be adapted to the CD approach. Using cross-domain competencies amongst team members to ensure effective communication and integration with other corporate functions has also been stressed by studies conducted in non-embedded domains (Feitelson et al., 2013; Cukier, 2013). Feitelson et al. (2013) highlight that at Facebook, software developers are trained to possess multiple skills, such as abilities in testing and operations, in order to ensure good-quality software all the time without the need for a supporting QA function (Feitelson et al., 2013). Similarly, in Cukier (2013), software developers also perform activities related to operation functions, in addition to performing QA functions activities. However, cross-domain competence may be more challenging in the embedded domain, where extensive knowledge of hardware-related aspects is needed. Additionally, several studies have proposed the use of common practices and platforms for software development and maintenance/operation/release as mechanisms to integrate corporate functions with the R&D organization (Brown et al., 2013; Fitzgerald and Stol, 2014; Gotel and Leip, 2007). Using shared models such as roadmaps and a feature dependency matrix has also been identified as a mechanism that facilitates effective coordination and interrelationships between solution designers and release managers of complex services (Ludwig et al., 2014).

5.10.2. Transparency

Organizational transparency, which is mentioned above as one of the main targets of integrated corporate functions, is a predecessor for building CD into an organization. Organizational transparency intends to show the bigger picture of scattered development activities in different parts of the organization, building a common understanding among stakeholders about the development progress and goals (Krusche and Alperowitz, 2014; Krusche et al., 2014; Gotel and Leip, 2007; Bellomo et al., 2013a; Ludwig et al., 2014; Poppendieck and Cusumano, 2012; Kalantar et al., 2014; Feitelson et al., 2013; Fitzgerald and Stol, 2014). In CD, transparency has an important role in creating the ability to foresee, trace and understand important aspects of product development in real time (Krusche et al., 2014; Neely and Stolt, 2013). Thus, transparency is also an enabler for identifying early risks that may harm product delivery and for reacting proactively to these risks. For example, it is important to provide mechanisms for visibility in the sales and marketing activities so that it is possible to track the status of the work in real time allowing for appropriate planning activities (Neely and Stolt, 2013). As described in Section 3.5.3, the importance of visualizing testing activities from end to end is highlighted in the literature (Nilsson et al., 2014).

In addition, it is noted that making the development progress transparent enables better awareness for team members oftheir contribution on the delivery of value (Staron et al., 2012), as well as allowing them to decide what is needed (Middleton and Joyce, 2012) and to take personal responsibility (Marschall, 2007). An important way for building transparency in the context of CD is the use of key performance indicators (KPIs) as metrics to visualize the performance of the organization. For instance, Staron et al. (2012) conducted a case study on a large agile and lean software development project at Ericsson in Sweden, in which KPIs were used to visualize the release readiness across the distributed teams in order to predict the time in weeks to release the product. Visualizing the quality of the outcomes by KPIs provided a way to build common awareness of the status among all stakeholders, from designers to unit managers (Staron et al., 2012). Therefore, it is important to identify useful metrics in the context of CD when building transparency. Other approaches for building transparency into an organization are proposed in the CD literature, e.g. traffic lights visualization is commonly used to display the status of production at any time (Krusche and Alperowitz, 2014). Moreover, information radiators and Kanban boards are often used around the work space to ensure that daily progress on a project is completely transparent and available for all to see (Middleton and Joyce, 2012).

5.10.3. Innovative and experimental organizational culture

Many primary studies stress the importance of people-driven development and the need to create an innovative and exper-

imental organizational culture to enable CD. According to the primary studies, learning from experience is more important and beneficial than chastising those responsible for a failure. Because humans make errors, some distrust is natural, but attention should be focused on honest communication and learning from mistakes; this improves trust among stakeholders, which enables greater efficiency and also innovativeness. In CD, failures are treated as opportunities for improvement rather than as occasions for assigning blame (Feitelson et al., 2013). Marschall (2007) emphasizes the importance of the role of individuals in taking responsibility for completing their tasks when producing value to a customer. Personal responsibilities can be used even to substitute methodologies and formalized procedures created for blaming and self-protection, as these procedures have no place in a team of engineers willing to take responsibility for the entire system (Feitelson et al., 2013).

The study by Papatheocharous et al. (2014) highlights the importance of human factors as a basis for increasing the capabilities of continuous software engineering. In their study they identified that providing tools for developers to improve themselves, and designing and assigning work tasks based on personal qualities may lead to situations where team members are willing to accept more responsibility in managing themselves, and, thus, take responsibility for the project outcome.

Regarding innovation, innovations should be encouraged by breaking the routine with frequent activities aimed towards new innovations. There should be flexibility and breakout times in the daily routines. For instance, hackathons are commonly used in Facebook in order to encourage interaction among different organizational functions from engineers to financial, legal and other departments (Feitelson et al., 2013).

6. Analysis of reported benefits and challenges for continuous deployment (RQ3)

6.1. Benefits

The literature highlights several benefits from applying CD. The most referenced benefits are shorter time-to-market, increased customer satisfaction, continuous feedback, rapid innovation, narrower test focus, improved release reliability and quality and improved developer productivity. However, the strength and quality of evidence for these benefits is limited as many claims are based on industry reports (i.e. practitioners' perceptions) or discussed in non-empirical studies. Furthermore, in many cases, benefits are claimed, but no rational or more detailed explanation of the reasons for these benefits is provided in the papers. Nonetheless, the benefits found in primary studies are detailed in the following paragraphs.

The most immediate benefit of applying CD is *shorter time-to-market* through fast and frequent releases (Agarwal, 2011; Blotner, 2002; Feitelson et al., 2013; Goodman and Elbaz, 2008; Krusche and Alperowitz, 2014; Ludwig et al., 2014; Middleton and Joyce, 2012; Olsson et al., 2012, 2013; Trimble and Webster, 2013, 2012; Benefield, 2009; Lacoste, 2009; Marschall, 2007; Neely and Stolt, 2013; Poppendieck and Cusumano, 2012; Khomh et al., 2012; Lavoie and Merlo, 2013). For instance, Feitelson et al. (2013); Marschall (2007); Neely and Stolt (2013) shortened their delivery cycles from months or weeks to continuous flow or daily deliveries. Similarly, in the context of embedded systems, Trimble and Webster (2013, 2012) reduced their release cycles significantly from three months to three weeks. Shorter release cycles enable companies to constantly develop, learn and improve their offerings based on instant customer feedback (Olsson et al., 2013; Lacoste, 2009; Neely and Stolt, 2013; Poppendieck and Cusumano, 2012; Khomh et al., 2012) and thus, companies can quickly learn what customers value and focus on deploying relevant functionalities that meet customers' expectations (Poppendieck and Cusumano, 2012; Olsson et al., 2013, 2012). Shorter release cycles enable faster feedback about new features and bug fixes, which makes release planning slightly easier (short term vs. long term planning) (Khomh et al., 2012). Moreover, a higher number of releases provides more marketing opportunities for companies (Khomh et al., 2012).

CD has also been found to increase *customer satisfaction* and enable *continuous customer feedback*. CD allows continual product enhancement and immediate access to new features and bug fixes, which increases customer satisfaction (Agarwal, 2011; Blotner, 2002; Neely and Stolt, 2013; Trimble and Webster, 2013; Khomh et al., 2012). For example, Neely and Stolt (2013) reported, *'We received an email from a customer saying that they had noticed the defect fix and wanted to say a huge thank you for resolving a pain point in the application'*. According to Khomh et al. (2012), under rapid release model, users can adopt new versions of the product faster, bugs are fixed faster and users do not experience significantly more post-release bugs in comparison with the traditional release model.

In addition, customers can evaluate the enhancements and provide feedback immediately and in a continuous way (i.e. continuous customer feedback), which improves communication between the company and its customers (Gotel and Leip, 2007; Krusche and Alperowitz, 2014; Krusche et al., 2014; Ludwig et al., 2014; MacCormack, 2001; Neely and Stolt, 2013; Trimble and Webster, 2012; Olsson et al., 2013, 2012). Furthermore, closer interaction with customers enables enterprises to monitor and collect instant field data on their customers and the software's behaviour (Cukier, 2013; Feitelson et al., 2013; Fitzgerald and Stol, 2014; Benefield, 2009). The main advantage is that companies have the chance to rapidly sense, understand and improve their offerings based on actionable metrics and data (Ko et al., 2011; MacCormack, 2001).

Apart from customer feedback, continuous and immediate feedback from CI and an automated infrastructure helps to identify and resolve issues more rapidly (Feitelson et al., 2013; Goodman and Elbaz, 2008; Humble et al., 2006; MacCormack, 2001; Neely and Stolt, 2013; Fitzgerald and Stol, 2014). For instance, Goodman and Elbaz (2008) observed that the CI process shortens the feedback cycle time substantially. Similarly, Humble et al. (2006) also reported that build and deployment scripts accelerate rapid feedback not just on the integration of modules of source code but also on problems integrating with the deployment environment and its external dependencies.

Closer relationships with customers further can facilitate *rapid innovation*. Continuous and instant customer feedback allows companies to invest their resources in developing relevant functionalities and innovation initiatives (Feitelson et al., 2013; Goodman and Elbaz, 2008). Olsson et al. (2012, 2013) observed that faster feedback means cheaper development since the R&D organization can then spend time developing the right things rather than correcting mistakes in functionality.

Narrower test focus appears also in our primary studies (Comas et al., 2011; Feitelson et al., 2013; Neely and Stolt, 2013; Mantyla et al., 2013). From a technical point of view, CD implies that each deployment introduces only limited amounts of new code. From this perspective, frequent releases with a smaller scope reduces risk (Neely and Stolt, 2013; Feitelson et al., 2013) and provides a narrower test focus, which more accurately guides quality assurance activities (Comas et al., 2011; Feitelson et al., 2013; Neely and Stolt, 2013; Mantyla et al., 2013). A narrow scope further allows deeper investigation of the product's active parts and makes issues easier to fix and debug (Comas et al., 2011; Feitelson et al., 2013; Neely and Stolt, 2013; Mantyla et al., 2013).

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

Several studies (Agarwal, 2011; Humble et al., 2006; Krusche and Alperowitz, 2014; Neely and Stolt, 2013; Benefield, 2009) also reported that the deployment infrastructure, coupled with intensive automated testing and fast rollback mechanisms, *improves release reliability and quality*. Neely and Stolt (2013) reported that automated deployment along with scrutiny of monitoring systems provided safer environments for shipping code. Furthermore, intensive automated tests ensure that new improvements pass all quality assurance procedures, thus leading to a higher quality of releases (Agarwal, 2011; Benefield, 2009; Neely and Stolt, 2013). Finally, automated deployment processes are reported as also leading to *improve developer productivity* since they allow a single engineer to develop and deploy a new improvement instantly to several services, to verify immediately and rollback to the previous stable version, if required (Agarwal, 2011; Gotel and Leip, 2007; Humble et al., 2006; Benefield, 2009).

6.2. Challenges

Regarding challenges in CD, the process of transformation towards CD, customers' unwillingness to receive continuous product updates, increased QA efforts and the challenges of applying CD in embedded domains were often referenced in the literature.

Transforming towards CD is an evolutionary process and requires investment in deployment processes, as well as changes in people's mindset and the general organization way of working. For example, Neely and Stolt (2013) describe how months of preparatory work were needed to get the deployment process streamlined and automated. Neely and Stolt (2013); Brown et al. (2013) observed that, if an organization is not experienced in ASD, a direct transition to CD requires too many changes to handle at one time. Furthermore, this transition requires a change in mindset as people may be afraid to release the new code directly into production environments (Marschall, 2007; Neely and Stolt, 2013). One company manager in Marschall (2007) noted, *'When the release team and I confronted the developers with our new process, releasing a story as soon as it is signed off it scared the hell out of them'*. Moving to CD requires a change in organizational culture, buy-in from all key stakeholders and transparency in the organization (Blotner, 2002; Middleton and Joyce, 2012; Neely and Stolt, 2013; Olsson et al., 2012). Lavoie and Merlo (2013) point out that fast release cycles might stress third-party developers because of the risk of non-compatibility of extension modules. Thus, human factors, including personality and cognitive aspects, plays a fundamental role in truly achieving continuous delivery. As acknowledged by Papatheocharous et al. (2014), in the context of continuous software engineering where organizations are required to develop, deliver and learn in fast and parallel cycles, it is profoundly important to establish an agile thinking culture from the individuals, to teams as well as upper management levels.

Even though customers seem to be more satisfied (see the previous section on benefits), the literature notes *customer unwillingness* to accept CD as another challenge (Olsson et al., 2013, 2012; Blotner, 2002; Agarwal, 2011; Orso et al., 2002). According to Agarwal (2011), typically, customers are reluctant to accept new functionalities mainly because of poor quality of releases. Orso et al. (2002) also identified privacy and security concerns about information collected from customers as inhibitors of CD. To address privacy concerns, which also suits monitoring purposes, they suggested that organizations should seek permission from users to gather information. One other inhibitor from a customer point of view is the learning curve that continuous changes (either to the functionality or to the user interface) request from the end-user perspective. Zade and Choppella (2012) empirically investigated the impact of changes on end-users. Their results suggest that the learning gap that changes in user interfaces may produce is a crucial factor to be considered when changes are continuously delivered to end-users.

Several studies (Agarwal, 2011; Kalantar et al., 2014; Krusche and Alperowitz, 2014; Marschall, 2007; Meyer et al., 2013; Neely and Stolt, 2013) reported *increased QA efforts* due to difficulties in managing the test automation infrastructure. Similarly, a case study conducted on Mozilla Firefox (Mantyla et al., 2013) found that while rapid release has numerous benefits and strongly supports shorter release times, at the same time it increases the test efforts. This stems from the fact that more specialized testers are required to sustain the testing effort in a rapid release model. In addition, CD requires establishing an effective QA process and new mechanisms to ensure backward compatibility of enhancements.

In addition, in the context of the *embedded domain*, Ard et al. (2014) reported that physical assets and hardware equipment should also support automation, in general, and CI, in particular, to get benefits from CD as a whole. Bosch and Eklund (2012) reported challenges concerning experimentation in software-intensive embedded systems. In particular, the architecture of embedded devices must support mechanisms to add or exchange applications when running experiments with minimal impact on the rest of the system. More importantly, the memory and processing footprint, as well as connectivity aspects, need to be considered carefully. Infrastructure requires the support of rollback mechanisms and immediate reversion to safe versions. Furthermore, since, in experimental scenarios, the infrastructure needed to keep track of individual devices, security and privacy issues require extra consideration (Bosch and Eklund, 2012). Finally, Trimble and Webster (2012) elaborated in the domain of mission critical systems. Safety issues require updates be planned and this prevents near real-time value delivery (e.g. users need to be notified and trained for new capabilities beforehand, to use them in mission-operation environments).

Finally, other challenges found in the literature, although they do not appear very frequently, include: lack of trust in software quality (Olsson et al., 2013), difficulty in managing various configurations and run-time environments (Kalantar et al., 2014; Meyer et al., 2013) and natural tensions between the desire to deliver functionalities quickly and the need for reliable products (Bellomo et al., 2013b; Brown et al., 2011). In addition, further difficulties exist in release planning and managing the roadmap in a fast-paced environment (Ludwig et al., 2014) and risks associated with gathering user feedback from a limited population (i.e. minority) that may constrain the software's evolution or even mislead product development (Ko et al., 2011).

7. Research gaps and opportunities for future research (RQ4)

The topic of CD appears to be highly relevant to the industry. Practitioners have contributed heavily to its body of knowledge, and the results of the quality assessment demonstrate the great significance of CD to the industry. However, with regard to empirical rigour, the quality assessment results are low (see Section 4.4). In addition, looking at the pertinence facet (see Table B.10, Appendix B), less than half of the studies (22 papers) are entirely dedicated to CD. Six of these produced contributions in the form of advice and implications, and three in the form of lessons learned. This is not necessarily negative as CD encompasses many different aspects. Still, more research that fully focused on CD is needed. In general, although the topic appears to be very promising, research on CD seems to be still in its infancy, which promises a range of new opportunities for researchers.

Each of the 10 identified themes represents opportunities for future research. However, the themes are explored at different levels, offering different research opportunities. The research to date

has tended to focus on factors such as continuous testing and QA (31 papers), fast and frequent release (28 papers), automation (24 papers, mainly in CI), and agile and lean software development (22 papers). However, only a small number of studies have dealt with aspects such as flexible design and architecture (9 papers), continuous and rapid experimentation (10 papers), and customer involvement (12 papers) in the context of CD. More concrete opportunities for future research include:

- Continuous and rapid experimentation is an emerging research topic with many possibilities for future work. Most papers in this area offer theoretical proposals that have yet to be adequately validated (e.g. Bosch and Eklund, 2012; Fagerholm et al., 2014). In particular, technical challenges (Bosch and Eklund, 2012), challenges with large scale experimentation (Bosch and Eklund, 2012; Fagerholm et al., 2014), business implications of continuous and rapid experimentation (Bosch and Eklund, 2012), privacy issues when running experiments with customers (Bosch and Eklund, 2012), and the process of transforming towards continuous experimentation itself (Olsson et al., 2012) are identified as areas for future research.
- Technical infrastructure for supporting CD. Incremental deployment is referred to in some of the industry reports in companies such as Facebook (Goel et al., 2014) and Rally (Neely and Stolt, 2013). However, how it is done in practice is unclear. Although some techniques and tools are mentioned in the literature, such as canary deployment and dark deployment (Feitelson et al., 2013; Neely and Stolt, 2013), they are briefly introduced without going into much detail on how they are actually used in practice. This topic is especially relevant when considering the importance of system availability and quality aspects related to the version of the system that is deployed at any given moment. Another example is Scuba as a solution to support monitoring of post-deployment user behaviour at Facebook (Goel et al., 2014). Although, the Scuba database is briefly introduce, however the kind of data that is collected as well as mechanisms to analyse it and feed it back to the development process are not described. Moreover, why certain technologies are selected over other similar existing solutions in the market has not been analysed. Thus, which are the most suitable technologies under certain conditions is unclear.
- Although customer involvement is emphasized in many primary studies, we discovered that the tasks required for making continuous and effective use of customer feedback are underdeveloped. Besides the needs for mechanisms to identify representative customers and infrastructure for collecting customer data, a clear research gap appears in solutions for processing incoming feedback and quickly interpreting the information. Further investigation is needed on new approaches to express and validate assumptions from user feedback, as the software evolves (Ko et al., 2011), and privacy and security concerns that may inhibit customer feedback collection (Orso et al., 2002).
- Regarding flexible and robust software architecture, some mechanisms for balancing stability and speed have been proposed in the literature (e.g. continuous assessment of quality attributes through prototyping or automated approaches and rapid architecture trade-off analysis). Nonetheless, the natural tension between the desire to deliver functionalities quickly and the need for reliable products is still a challenge for CD (Bellomo et al., 2013b; Brown et al., 2011). Investigating measurement systems for managing technical debt (Bellomo et al., 2013b), risk assessment methods for CD (Comas et al., 2011), and stability-triggers-speed scenarios (i.e. stability causes a refocus on speed) (Bellomo et al., 2013b) are areas proposed by the primary studies for future research.
- Continuous planning also seems to play an important role in achieving CD. Based on our results, continuous planning is not commonly adopted and applied throughout the entire organization. It is currently connected to prioritization (Trimble and Webster, 2013) or involves only a certain level of planning, mainly release planning (e.g. Continuous Scrum Agarwal, 2011). Fitzgerald and Stol (Fitzgerald and Stol, 2014) found that the current focus of continuous planning in CD is mainly on what emerges from ASD, which it is related to sprint iterations or, at best, software releases. Thus, empirical research rarely describes how continuous planning is conducted at different organizational levels and how the information from plans is visible at different levels of planning.
- Similarly, despite the identified need for, and importance of, integrating R&D with other corporate functions, there is very little empirical research that evaluates emerging approaches such as DevOps and BizDez (Fitzgerald and Stol, 2014). Research is needed to identify and empirically evaluate mechanisms to facilitate collaboration between different organizational functions, not just at the organizational level but also at the technical level.
- We also believe that automation is an important area for future research. The goal of automation is to eliminate all manual steps from build to deployment processes. The literature shows different research gaps regarding automation. For example, GUI testing (Agarwal, 2011) is identified as a lead indicator of bugs that typically appear in production environments. However, GUI testing still requires manual observation because of difficulties in automatically identifying errors. This is an area for possible future research. In particular, Agarwal (2011) pointed out the possibility of utilizing advanced techniques in video analytics to enable fully automated GUI testing. Similarly, dealing with variability is considered an area that calls for future work, particularly when automating the deployment of software components (Van Der Storm, 2005).
- The application of CD in contexts that are different from web applications (i.e. embedded systems) presents a clear opportunity for future research as different domains have different constraints when applying CD. As identified in Section 4, when reviewing the study domain, the majority of studies have been conducted in the web application domain. Organizations in the area of web applications are currently able to implement CD and even deploy many new versions per day (e.g. Feitelson et al., 2013; Marschall, 2007; Neely and Stolt, 2013). However, this goal is still a formidable challenge for the systems and services domains, such as in embedded systems. Although the application of CD in embedded systems has attracted significant interest over the last four years (see Fig. 7), many challenges are still apparent for CD in embedded systems. For example, in the area of post-deployment activities, those primary studies that have used or discussed post-deployment activities focus on cloud- or web-based domains. However, unlike web-based companies, not all software development companies have access to a huge amount of customers. Another example is the case of software ecosystems, which are becoming increasingly popular. The implications of using CD with different contributors or when interdependent systems need to be coordinated are not clear.
- The implications of human aspects with CD are also underdeveloped. With the adoption of CD, individuals are given more responsibility for systems under development. For example, in an optimal situation, an individual developer could deploy a new version of the system directly to customers in just a couple of automated steps. However, the individual should be also ready to take on that responsibility. Papatheocharous et al. (2014) argue, *'models considering the individuals personal traits*

to accommodate the objective of holistic continuous software engineering' are scarce. Models such as Stairway to Heaven (Olsson et al., 2012) provide guidance on organizational levels, but how CD relates to personality and cognitive factors remains an unexplored research area.

To conclude, besides specific areas that require future research, we find that CD needs an increase in both the number and rigour of empirical studies. Recent case studies, such as those that focus on Mozilla Firefox (Baysal et al., 2012; Khomh et al., 2012; Lavoie and Merlo, 2013; Mantyla et al., 2013) or the experiment conducted by Zade and Choppella (2012), provide rigorous contributions to illustrate the impact of CD on product/process quality as well as on the end customer. However, these studies explicitly mention limitations in generalizing results. Thus, similar studies are needed. Industry participation is essential in order to obtain the right data set that allows to produce reliable results (particularly in research aspects that are the hardest to research in terms of data availability such as those that involve customers). Moreover, although several benefits related to CD are mentioned in the literature, they are mainly referred to in industry reports and non-empirical research. Thus, empirical studies and scientific evidence that confirm or refute these benefits as well as studies that analyse cause-effect relationship between identified factors might be investigated in future work. Similarly, emerging approaches for implementing CD require more rigorously empirical evaluations that can help to generalize results where possible. In particular, the lack of context descriptions in the primary studies makes comparing different studies and providing more generalizable results difficult.

8. Comparison to related reviews

In order to put the findings of this work into the context of literature and highlight the contributions made by the present work, we compare our results with related literature reviews as presented in Section 2.2. Since ASD and specific agile practices are not the centre of our work, the comparison focuses mainly on our findings and the findings of the semi-systematic literature study conducted by Mäntylä et al. (2014).

Both studies differ slightly in terms of focus and scope. Still, the benefits and challenges of CD, are at the heart of both reviews. The focus of Mäntylä et al. (2014) is mainly on the impact of rapid releases on testing processes; additionally, the results are extended with a semi-systematic literature review, which includes 24 primary studies, to investigate the prevalence of rapid release, spanning its origination, enablers, benefits and problems. Our study, on the other hand, seeks to collect and synthesize all relevant studies on the topic of CD in a systematic manner in order to characterize this phenomenon by identifying recurrent themes and exploring the multifaceted nature of CD. We found 50 primary studies relevant in the context of CD. Our study extends Mäntylä et al.'s semi-systematic literature review in that its focus is entirely on CD and its recurrent themes; it includes scientific studies published up to June 2014 (comprising theoretical studies as well), and takes into account all aspects of the systematic literature review method in order to ensure reliable results, such as peer reviews of each research step, systematic data extraction and data synthesis, and quality assessment of primary studies. These aspects are absent in the study conducted by Mäntylä et al. (2014).

In a similar vein to Mäntylä et al. (2014), we found that CD has had an important impact on the software industry in recent years (see Fig. 8). Indeed, many of our primary studies, especially those published in recent years, discuss specific cases of real companies, such as Facebook (Goel et al., 2014; Feitelson et al., 2013), Firefox (Baysal et al., 2012; Khomh et al., 2012; Lavoie and Merlo, 2013; Mantyla et al., 2013), IBM (Cantor and Royce, 2014; Gotel and Leip, 2007; Kalantar et al., 2014; Ludwig et al., 2014; Brown et al., 2013), Rally Corporation (Neely and Stolt, 2013), Ericsson (Staron et al., 2012; Antinyan et al., 2014), Volvo (Antinyan et al., 2014) or NASA (Trimble and Webster, 2013, 2012). The fact that CD has been adopted in such a diverse set of companies provides testimony that CD is being applied in many different domains. Our primary studies include evidence of CD being adopted even in domains such as safety critical (e.g., Trimble and Webster, 2013, 2012; Ard et al., 2014) and science systems (Wicenec et al., 2012). Thus, our results confirm the claim made by Mäntylä et al. (2014) that CD can be part of any software development domain. In addition, our study presents an overview of the state-of-the-art of CD (Section 4), which, as presented in Section 7, reveals that the body of knowledge of CD is in an exploratory stage and its scientific evidence must be improve.

Mäntylä et al. (2014) also discuss enablers as *'accelerating factors that facilitate the adoption of rapid releases.'* We did not specifically focus on enablers as preconditions of CD, as we had a wider scope to investigate recurrent factors in the literature related to CD. Mäntylä et al. (2014) found that *'parallel development with tools enabling easy automatic deployment and testing, and with proactive customers and product managers'* are enablers of rapid releases. However, the actual mechanisms to achieve frequent releases, tools employed to support automation, and strategies for involving customers in the product development process were not further elaborated. Our study extends Mäntylä et al.'s preliminary identification of important aspects enabling rapid software releases by creating a schema that classifies recurrent aspects in the CD literature and identifies concrete frameworks, methods and tools that support CD in practice.

Our classification schema comprises 10 underpinning factors that define CD. Factors already identified in the study by Mäntylä et al. remain in our classification schema under the following themes: fast and frequent releases, automation, continuous testing and quality assurance, and customer involvement. However, we also found other aspects that are relevant in the context of CD, such as flexible product design and architecture, configuration management, continuous and rapid experimentation, post-deployment activities, agile and lean software development and organizational factors, such as integrated corporative functions, transparency, and the need of creating an innovative and experimental organizational culture. Indeed, these factors are not independent of each other, but have overlaps and synergies between them that help support CD in practice. For example, continuous testing and quality assurance is critical in CD because the product is continuously deployed to the end customer. Thus, mechanisms that assure quality, such as automated testing, are essential in the context of CD. However, automation is much more than just automating the testing process; it also aims to minimize the manual overhead by automating the entire end-to-end workflow, including aspects such as automation of the delivery process, configuration management, etc. In a similar vein, experimentation, as a way of making decisions based on objective data rather than 'gurus' opinions that may lead to incorrect interpretations of the reality, is emphasized in the CD body of knowledge. Moreover, it is considered to be a common-sense approach to proactively involve customers and customers' views in the development process.

Similar to the findings of Mäntylä et al., we found that tool support is very important. Table B.11 in Appendix B summarizes the concrete tools and methods that, according to our primary studies, software intensive companies use to apply CD. Moreover, our findings are aligned with the claim that *'release length simply appeared as a variable in the release models without providing insights regarding rapid releases'* (Mäntylä et al., 2014). Thus, the release length

in our primary studies varied from a few hours to a few weeks, depending mainly on the application domain.

With regard to the benefits and challenges of using CD, our findings are well aligned with the results already reported by Mäntylä et al. (2014). Shorter time-to-market, increased customer satisfaction, continuous rapid feedback, and narrower test focus were among the benefits identified in both reviews. According to the primary studies in our review, developers are more productive when using CD as a direct consequence of the automation of deployment processes. According to the review conducted by Mäntylä et al., efficiency also increases, but as a result of time-pressure. In addition, we found that the deployment infrastructure supporting on automation and fast rollback mechanisms improves release reliability and quality. Mäntylä et al. note that using CD makes it easier to monitor progress and quality.

Finally, with regard to the challenges of using CD, we found four main aspects were reported in the literature: the process of transforming towards CD, customers' unwillingness to accept continuous updates, increased QA efforts, and the application of CD in the embedded domain. Although Mäntylä et al. found customer unwillingness to be a challenge with rapid releases, the challenges identified in their review mainly focus on testing areas, such as conflicting goals between rapid release and achieving high reliability and test coverage. Still, as described above, these findings need to be confirmed through more rigorous scientific studies as, at the moment, they are mainly based on practitioner perceptions.

9. Conclusion

This study provides a structured understanding of the body of CD knowledge, together with a systematically collected list of references relevant to CD. By using the systematic mapping method, we identified, classified and analysed primary studies related to CD based on a survey of the literature conducted in June 2014. The most important findings of this review, which are organized according to the study's research questions, are summarized below.

- *RQ1: What is the current state of the research pertaining to CD in the context of software intensive products and services?* From the 21,382 retrieved documents, we identified 50 primary studies relevant to CD. Most of these primary studies are industry reports (36%) and case studies (24%). Other research methods that were used included action research (4%) and grounded theory, mixed method, design science and experiment, each with just 2% of the studies. Overall, 42% of the primary studies contributed to CD in the context of web/Internet-based services and applications, and 24% were related to embedded systems. Eight percent of the studies focused on desktop applications and the domains of the remaining 26% of the primary studies were not clearly described. In addition, 8% of primary studies contributed to CD theoretically (without empirical evidence). A rigour and relevance analysis indicated that 37 primary studies exhibited high relevance; however, of these, only 14 studies showed high rigour and 23 studies had less than moderate rigour. This provides clear evidence that scientific contributions in the literature on CD are currently of high relevance but medium-low rigour, which calls for more meticulous research.
- *RQ2: What are the main factors that characterize CD in the context of software intensive products and services (sub: what do researchers mean when they refer to CD)?* A general consensus exists among most authors of the literature surveyed that CD refers to the ability of organizations to release software functionalities to customers quickly and frequently, soon after each new functionality is developed. However, these authors tended to use the concept interchangeably with continuous delivery (although as it is described in Section 2.1, they have different meanings).
  As a result of our analysis of the 50 primary studies, we identified 10 recurrent themes or factors related to CD. Each of the 10 factors was analysed and presented in detail in Section 5. The factors include: (1) fast and frequent release, (2) flexible product design and architecture, (3) continuous testing and quality assurance, (4) automation (of build and test (CI), deployment/delivery/release processes and configuration of deployment environments), (5) configuration management, (6) customer involvement, (7) continuous and rapid experimentation, (8) post-deployment activities, (9) agile and lean software development and (10) organizational factors, including integrated corporate functions, transparency and an innovative and experimental organizational culture.
- *RQ3. What are the reported benefits and challenges in association with CD in the context of software intensive products and services?* A number of benefits and challenges related to CD were identified. Transforming towards CD is identified as challenging, requiring significant investment in deployment processes, as well as in changes in people's mindsets and organizations' general way of working. In addition, an unwillingness by some customers to accept new functionality, a need to increase efforts in QA and the application of CD in the context of embedded systems were identified as significant challenges. However, CD also poses potential benefits for organizations, such as shortening their time-to-market by reinforcing the organizations' capabilities to release software functionalities to customers more quickly and frequently, an increase in customer satisfaction with the continual deployment of valuable product enhancements and obtaining immediate feedback during the development process, particularly from customers, which helps to guide software development activities and quickly identifies potential problems. In addition, CD also appears to facilitate rapid innovation through experimentation and continuous and instant customer feedback and to improve release reliability and quality, in part, due to a narrower test focus and the extensive use of automation. However, these findings have to be carefully interpreted, as the empirical evidence is limited mainly to practitioners' perceptions.
- *RQ4. What are the research gaps in the area of CD in the context of software intensive products and services?* Finally, a plethora of venues for future research, due to the topics freshness and its industrial relevance were identified. Rigorous scientific contributions are clearly needed, particularly those based on empirical evidence evaluating the benefits of CD. In addition, we identified a number of research gaps within the 10 themes identified. Continuous and rapid experimentation is an emerging research topic with many avenues for future work. Similarly, a clear research gap exists for mechanisms to use customer feedback in the most appropriate way so that information can be quickly interpreted. In continuous testing and QA, research contributions on mechanisms for implementing automated GUI testing are required, as well as investigations assessing technical debt in the context of CD. In addition, topics such as continuous planning, automation and integrated corporate functions (e.g. DevOps and BizDez), all appear to be especially relevant. In addition to the above specific areas of future research, CD research needs to increase in both number and, especially, rigour of empirical studies.

Acknowledgement

This research has been carried out within the Digile Need for Speed program, and it has been partially funded by Tekes (the Finnish Funding Agency for Technology and Innovation).

Appendix A. Extracted data – primary study properties

Table A.9
Extracted data – primary study properties.

Category	Description
P1: General type of paper (adapted from Kitchenham, 2010).	
Empirical	If the paper bases its findings on empirical evidence (exploration of the phenomenon of CD, explanation of some aspect related to CD, evaluation of a CD technique, etc.). Therefore, the source of knowledge of the paper is acquired by means of observation or experimentation. Observations can be carried out by using different data collection methods such as direct observation of the phenomenon or interviews, surveys, focus groups, etc.
Theoretical	If the paper is descriptive and discusses some issue (theories, frameworks, or underlying concepts) and may (but not always) consider some theoretical issues concerning CD. It does not include an empirical study of the issue being discussed. Typically tools and frameworks that ae not empirically validated as well as conceptual and mathematical analyses are included in this category.
Both	The paper is a mixed theoretical and empirical paper. Typically papers that development techniques or frameworks with the intention of CD and provide some empirical evaluation or demonstration of the technique are included in this category.
P2: Research method (adapted from Unterkalmsteiner et al., 2012).	
Case study	If one of the following criteria applies: (1) The study declares one or more research questions which are answered (completely or partially) by applying a case study. (2) The study empirically evaluates a theoretical concept by applying it in a case study (without necessarily explicitly stating research questions, but having a clearly defined goal).
Industry report	If the focus of the study is directed toward reporting industrial experiences without stating research questions or a theoretical concept which is then evaluated empirically. Usually these studies do not mention any research method explicitly.
Experiment	If the study conducts an experiment and clearly defines its design (variables, control group, treatment, etc.).
Survey	If the study collects quantitative and/or qualitative data by means of a questionnaire or interviews. In a survey study a sample that is representative of the population is studied in order to generalize results from the sample to the whole population (opposite to a case study in which only one or a limited number of cases is considered).
Action research	If the study states this research method explicitly.
Design science	If the study states this research method explicitly.
Grounded theory	If the study states this research method explicitly.
Mixed method	If the study uses multiple methods for data collection.
Not stated	If the study does not define the applied research method and it cannot be derived or interpreted from reading the paper.
Opinion paper	If the paper expresses the personal opinion of an author about CD or whether a certain aspect of CD is good or bad, or how it should be applied. The paper does not rely on related work and research methods and does not explicitly describe industrial experiences.
P3: Contribution (adapted from Paternoster et al., 2014 and Shaw, 2003).	
Model	Representation of an observed reality by concepts or related concepts after a conceptualization process.
Theory	Construct of cause-effect relationships of determined results.
Framework/Method	Method or technique related to constructing software or managing development processes. Commonly it involves better ways to do some task.
Guidelines	List of advises, synthesis of the obtained research results.
Lesson learned	Set of outcomes, directly analysed from the obtained research results.
Advice/Implications	Discursive and generic recommendation, deemed from personal opinions.
Tool	Technology, programme or application used to create, debug, maintain or support software development processes.
P5: Pertinence (inspired by Paternoster et al., 2014).	
Full	The main focus of the study is CD. The three characteristic of CD are noticeable in the study (deployment, continuity and speed).
Partial	The study is partially related to CD. The study supports CD or focuses on an aspect that is important in the context of CD but CD as a whole is not its main focus.
Marginal	The study is marginally related to CD. CD is mentioned in the study but the main research focus of the study is different from CD.

Appendix B. Systematic map overview

Table B.10
Systematic map overview.

PS	Research method	Contribution	Domain	Pertinence	Rigour	Relev.
Agarwal (2011)	Industry report	Framework/method	Web/services	Full	1.5	4
Bellomo et al. (2013b)	Grounded theory	Theory	Multiple domains	Partial	2.5	4
Cantor and Royce (2014)	Industry report	Framework/method	N/S	Marginal	1	4
Comas et al. (2011)	Theoretical	Framework/method	Web/services	Partial	2	0
Cukier (2013)	Industry report	Guidelines	Web/service	Full	1.5	4
Feitelson et al. (2013)	Industry report	Advice/implications	Web/Service	Full	1.5	4
Krusche and Alperowitz (2014)	Case study	Lesson learned	N/S	Full	3	0
Ludwig et al. (2014)	Theoretical + industry report	Framework/method	Web/service	Partial	0.5	3
MacCormack (2001)	Mixed methods	Lesson learned	Web/service	Full	2	4
Marschall (2007)	Industry report	Advice/implications	Web/service	Full	1.5	4
Neely and Stolt (2013)	Industry report	Lesson learned	Web/service	Full	1	4
Trimble and Webster (2012)	Industry report	Lesson learned	Embedded, safety critical	Partial	0.5	4
Trimble and Webster (2013)	Industry report	Lesson learned	Embedded, safety critical	Partial	1	4
Van Der Storm (2005)	Theoretical + case study	Framework/method	N/S	Partial	1	0
Brown et al. (2013)	Industry report	Framework/method	N/S	Full	1	4
Brown et al. (2011)	Theoretical + case study	Framework/method	Embedded	Partial	1	2
Bellomo et al. (2013a)	Industry report	Guidelines	Web/service	Partial	1.5	4
Poppendieck and Cusumano (2012)	Opinion paper	Advice/implications	N/S	Partial	N/A	N/A
Antinyan et al. (2014)	Action research	Framework/method	Embedded	Partial	3	4
Blotner (2002)	Industry report	Advice/implication	Web/service	Marginal	1	4
Bosch and Eklund (2012)	Theoretical + case study	Model	Embedded	Full	1	3
Fagerholm et al. (2014)	Design science	Model	Web/service	Full	2	3
Fitzgerald and Stol (2014)	Theoretical	Model	N/S	Full	2	0
Goodman and Elbaz (2008)	Industry report	Advice/implications	Web/service	Full	0.5	4
Gotel and Leip (2007)	Industry report	Advice/implications	Web/service	Full	1.5	4
Humble et al. (2006)	Industry report	Guidelines	Web/service	Full	1	4
Kalantar et al. (2014)	Industry report	Tool	Web/service	Full	1	4
Ko et al. (2011)	Case study	Lesson learned	Web/service	Partial	2.5	4
Krusche et al. (2014)	Theoretical + mixed methods	Model	N/S	Full	3	0
Lacoste (2009)	Industry report	Advice/implications	Web/service	Partial	1.5	4
Meyer et al. (2013)	Theoretical	Tool	N/S, open source	Partial	1.5	0
Middleton and Joyce (2012)	Case study	Lesson learned	Web/service	Partial	3	4
Olsson et al. (2012)	Case study	Model, lesson learned	Embedded	Full	3	4
Olsson et al. (2013)	Case study	Model, lesson learned	Embedded and web/services	Full	3	4
Orso et al. (2002)	Theoretical + case study	Tool	N/S	Partial	1	1
Scacchi and Alspaugh (2013)	Case study	Lesson learned	N/S	Marginal	1	1
Wicenec et al. (2012)	Industry report	Advice/implications	Embedded	Full	0.5	3
Eklund and Bosch (2012)	Theoretical + case study	Model	Embedded	Full	1	3
Ard et al. (2014)	Theoretical + industry report	Lesson learned, model	Embedded	Partial	1	4
Nagy et al. (2010)	Theoretical	Model	N/S	Partial	2	0
Benefield (2009)	Opinion paper	Advice/implications	Web/service	Full	N/A	N/A
Khomh et al. (2012)	Case study	Lesson learned	Desktop application	Partial	3	4
Goel et al. (2014)	Industry report	Framework/method	Web/service	Full	1.5	4
Lavoie and Merlo (2013)	Case study	Lesson learned	Desktop application	Partial	2	4
Mantyla et al. (2013)	Case study	Model	Desktop application	Partial	3	4
Staron et al. (2012)	Action research	Lesson learned, framework/method	Embedded	Partial	3	4
Nilsson et al. (2014)	Case study	Framework/method	Embedded	Partial	2.5	4
Papatheocharous et al. (2014)	Case study	Framework/method	N/S	Partial	1.5	4
Baysal et al. (2012)	Case study	Lesson learned, model	Desktop application	Partial	3	4
Zade and Choppella (2012)	Experiment	Theory	N/S	Marginal	3	1

Table B.11
Frameworks and methods, models and tools.

Primary study	Description	Pertinence
	Frameworks and methods	
Agarwal (2011)	Continuous SCRUM, an approach based on Scrum to achieve fast-paced continuous product evolution and deployment	Full
Goel et al. (2014)	Solution applied at Facebook to enable frequent software upgrades.	Full
Brown et al. (2013) and (Cantor and Royce, 2014)	Economic governance, measured improvement and disciplined agile delivery frameworks to accelerate software delivery at an enterprise scale	Full and marginal
Comas et al. (2011)	Risk assessment heuristic approach to quantify software deployment risks in the context of fast-paced continuous release environment	Partial
Antinyan et al. (2014)	Method to identify risky areas of source code and assess risks in the context of fast and incremental delivery environment	Partial
Brown et al. (2011)	Approach to quantify architecture quality with measurable criteria to guide continuous and iterative release planning	Partial
Van Der Storm (2005)	Approach to continuous release and upgrade of component-based software	Partial
Nilsson et al. (2014)	Visualization technique of the testing activities involved from unit and component level to product and release level that support the identification of improvement areas	Partial
Papatheocharous et al. (2014)	Two level approach of how human factors can influence continuous software engineering	Partial
Staron et al. (2012)	Release Readiness Indicator to predict the time in weeks to release the product	Partial
Ludwig et al. (2014)	Approach to manage dependences between service design and release management	Partial
	Models	
Olsson et al. (2012) and (Olsson et al., 2013)	Stairway to Heaven model	Full
Fagerholm et al. (2014)	Continuous experimentation model	Full
Bosch and Eklund (2012) and (Eklund and Bosch, 2012)	Architecture for large-scale innovation experiment system	Full
Fitzgerald and Stol (2014)	Continuous software engineering model	Full
Krusche et al. (2014)	Rubgy: Agile process model for continuous delivery	Full
Nagy et al. (2010)	Project health measurement model based on bayesian networks	Partial
Mantyla et al. (2013)	Model explaining the relationship between release model, release length and test effort	Partial
Baysal et al. (2012)	Model of the Mozilla's patch lifecycle for rapid releases	Partial
Ard et al. (2014)	Software in Simulation (SiS) architecture to practice continuous integration and continuous deployment in the embedded domain	Partial
	Tools	
Kalantar et al. (2014)	Weaver. A domain-specific language for continuous validation of the deployed environment	Full
Meyer et al. (2013)	Automated quality assurance service to validate configuration management scripts across a range of environments	Partial
Orso et al. (2002)	GAMMA. Tool for remotely monitoring of deployed software	Partial
Lavoie and Merlo (2013)	Clone detector as a tool to understand differences between releases such as how many changes were done between releases, how many bugs were made, etc.	Partial

References

Agarwal, P., 2011. Continuous Scrum: agile management of SaaS products. In: Proceedings of the 4th India Software Engineering Conference. ACM, pp. 51–60. *[PS1].

Antinyan, V., Staron, M., Meding, W., Osterstrom, P., Wikstrom, E., Wranker, J., Henriksson, A., Hansson, J., 2014. Identifying risky areas of software code in agile/lean software development: an industrial experience report. In: 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE). IEEE, pp. 154–163. *[PS2].

Ard, J., Davidsen, K., Hurst, T., 2014. Simulation-based embedded agile development. IEEE Software 31 (2), 97–101. *[PS3].

Baysal, O., Kononenko, O., Holmes, R., Godfrey, M.W., 2012. The secret life of patches: a Firefox case study. In: 2012 19th Working Conference on Reverse Engineering (WCRE). IEEE, pp. 447–455. *[PS4].

Beck, K., 2000. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al., 2001. The agile manifesto.

Bellomo, S., Nord, R.L., Ozkaya, I., 2013. Elaboration on an integrated architecture and requirement practice: prototyping with quality attribute focus. In: 2013 2nd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks). IEEE, pp. 8–13. *[PS5].

Bellomo, S., Nord, R.L., Ozkaya, I., 2013. A study of enabling factors for rapid fielding combined practices to balance speed and stability. In: 2013 35th International Conference on Software Engineering (ICSE). IEEE, pp. 982–991. *[PS6].

Benefield, R., 2009. Agile deployment: lean service management and deployment strategies for the SaaS enterprise. In: 42nd Hawaii International Conference on System Sciences, 2009, HICSS'09. IEEE, pp. 1–5. *[PS7].

Blotner, J.A., 2002. Agile techniques to avoid firefighting at a start-up. In: OOPSLA 2002 Practitioners Reports. ACM, pp. 1–ff. *[PS8].

Boehm, B., 2002. Get ready for agile methods, with care. Computer 35 (1), 64–69.

Boehm, B., 2006. A view of 20th and 21st century software engineering. In: Proceedings of the 28th International Conference on Software Engineering. ACM, pp. 12–29.

Boehm, B.W., 1988. A spiral model of software development and enhancement. Computer 21 (5), 61–72.

Bosch, J., 2012. Building products as innovation experiment systems. In: Software Business. Springer, pp. 27–39.

Bosch, J., Eklund, U., 2012. Eternal embedded software: towards innovation experiment systems. In: Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change, vol. 7609. Springer, Berlin, Heidelberg, pp. 19–31. *[PS9].

Brown, A.W., Ambler, S., Royce, W., 2013. Agility at scale: economic governance, measured improvement, and disciplined delivery. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, pp. 873–881. *[PS10].

Brown, N., Nord, R.L., Ozkaya, I., Pais, M., 2011. Analysis and management of architectural dependencies in iterative release planning. In: 2011 9th Working IEEE/IFIP Conference on Software Architecture (WICSA). IEEE, pp. 103–112. *[PS11].

Cantor, M., Royce, W., 2014. Economic governance of software delivery. IEEE Software 31 (1), 54–61. doi:10.1109/MS.2013.102. *[PS12].

Castells, M., 2011, vol. 1. The Rise of the Network Society: The Information Age: Economy, Society, and Culture. John Wiley & Sons.

Causevic, A., Sundmark, D., Punnekkat, S., 2011. Factors limiting industrial adoption of test driven development: a systematic review. In: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST). IEEE, pp. 337–346.

Claps, G.G., Svensson, R.B., Aurum, A., 2015. On the journey to continuous deployment: technical and social challenges along the way. Inf. Software Technol. 57, 21–31.

Comas, J., Mostashari, A., Mansouri, M., Turner, R., 2011. A software deployment risk assessment heuristic for use in a rapidly-changing business-to-consumer web environment. Int. J. Software Eng. Appl. 5 (4), 107–126. *[PS13].

Cruzes, D.S., Dyba, T., 2011. Recommended steps for thematic synthesis in software engineering. In: 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, pp. 275–284.

Cukier, D., 2013. DevOps patterns to scale web applications using cloud services. In: Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, & Applications: Software for Humanity. ACM, pp. 143–152. *[PS14].

Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B., 2012. A decade of agile methodologies: towards explaining agile software development. J. Syst. Software 85 (6), 1213–1221.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: a systematic review. Inf. Software Technol. 50 (9), 833–859.

Eck, A., Uebernickel, F., Brenner, W., 2014. Fit for continuous integration: how organizations assimilate an agile practice. In: Proceedings of the Twentieth Americas Conference on Information Systems. Savannah.

Eisenhardt, K.M., Martin, J.A., 2000. Dynamic capabilities: what are they? Strategic Manage. J. 21 (10–11), 1105–1121.

Eklund, U., Bosch, J., 2012. Architecture for large-scale innovation experiment systems. In: 2012 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA). IEEE, pp. 244–248. *[PS15].

Fagerholm, F., Guinea, A.S., Mäenpää, H., Münch, J., 2014. Building blocks for continuous experimentation. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (RCoSE 2014), Hyderabad, India. *[PS16].

Feitelson, D.G., Frachtenberg, E., Beck, K.L., 2013. Development and deployment at Facebook. IEEE Internet Comput. 17 (4), 8–17. *[PS17].

Fitzgerald, B., Stol, K.-J., 2014. Continuous software engineering and beyond: trends and challenges. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. ACM, pp. 1–9. *[PS18].

Goel, A., Chopra, B., Gerea, C., Mátáni, D., Metzler, J., Ul Haq, F., Wiener, J., 2014. Fast database restarts at Facebook. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. ACM, pp. 541–549. *[PS19].

Goodman, D., Elbaz, M., 2008. "It's not the pants, it's the people in the pants" learnings from the gap agile transformation what worked, how we did it, and what still puzzles us. In: Agile Conference, 2008, AGILE'08. IEEE, pp. 112–115. *[PS20].

Gotel, O., Leip, D., 2007. Agile software development meets corporate deployment procedures: stretching the agile envelope. In: Agile Processes in Software Engineering and Extreme Programming. Springer, pp. 24–27. *[PS21].

Humble, J., Farley, D., 2010. Continuous Delivery: Reliable Software Releases through Build, Test, and deployment automation. Pearson Education.

Humble, J., Read, C., North, D., 2006. The deployment production line. In: Agile Conference, 2006. IEEE, p. 6. *[PS22].

Ivarsson, M., Gorschek, T., 2011. A method for evaluating rigor and industrial relevance of technology evaluations. Empirical Software Eng. 16 (3), 365–395.

Järvinen, J., Huomo, T., Mikkonen, T., Tyrväinen, P., 2014. From agile software development to mercury business. In: Software Business. Towards Continuous Value Delivery. Springer, pp. 58–71.

Kalantar, M., Rosenberg, F., Doran, J., Eilam, T., Elder, M., Oliveira, F., Snible, E., Roth, T., 2014. Weaver: language and runtime for software defined environments. IBM J. Res. Dev. 58 (2), 1–12. *[PS23].

Khomh, F., Dhaliwal, T., Zou, Y., Adams, B., 2012. Do faster releases improve software quality? An empirical case study of Mozilla Firefox. In: 2012 9th IEEE Working Conference on Mining Software Repositories (MSR). IEEE, pp. 179–188. *[PS24].

Kitchenham, B., 2010. What's up with software metrics?–A preliminary mapping study. J. Syst. Software 83 (1), 37–51.

Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Technical report, EBSE-2007-01.

Kitchenham, B.A., Budgen, D., Pearl Brereton, O., 2011. Using mapping studies as the basis for further research–a participant-observer case study. Inf. Software Technol. 53 (6), 638–651.

Ko, A.J., Lee, M.J., Ferrari, V., Ip, S., Tran, C., 2011. A case study of post-deployment user feedback triage. In: Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering. ACM, pp. 1–8. *[PS25].

Krusche, S., Alperowitz, L., 2014. Introduction of continuous delivery in multi-customer project courses. In: Companion Proceedings of the 36th International Conference on Software Engineering. ACM, pp. 335–343. *[PS26]

Krusche, S., Alperowitz, L., Bruegge, B., Wagner, M.O., 2014. Rugby: an agile process model based on continuous delivery. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. ACM, pp. 42–50. *[PS27].

Lacoste, F.J., 2009. Killing the gatekeeper: Introducing a continuous integration system. In: Agile Conference, 2009, AGILE'09. IEEE, pp. 387–392. *[PS28].

Lavoie, T., Merlo, E., 2013. How much really changes? A case study of Firefox version evolution using a clone detector. In: Proceedings of the 7th International Workshop on Software Clones. IEEE Press, pp. 83–89. *[PS29].

Ludwig, H., Cappi, J., Becker, V., Stewart, B., Meade, S., 2014. Integrating service release management with service solution design. In: Service-Oriented Computing–ICSOC 2013 Workshops. Springer, pp. 28–39. *[PS30].

MacCormack, A., 2001. How internet companies build software. MIT Sloan Manage. Rev. 42 (2), 75–84. *[PS31].

Mäntylä, M.V., Adams, B., Khomh, F., Engström, E., Petersen, K., 2014. On rapid releases and software testing: a case study and a semi-systematic literature review. Empirical Software Eng. 1–42.

Mantyla, M.V., Khomh, F., Adams, B., Engstrom, E., Petersen, K., 2013. On rapid releases and software testing. In: 2013 29th IEEE International Conference on Software Maintenance (ICSM). IEEE, pp. 20–29. *[PS32].

Marschall, M., 2007. Transforming a six month release cycle to continuous flow. In: Agile Conference (AGILE), 2007. IEEE, pp. 395–400. *[PS33].

Merisalo-Rantanen, H., Tuunanen, T., Rossi, M., 2005. Is extreme programming just old wine in new bottles: A comparison of two cases. J. Database Manage. (JDM) 16 (4), 41–61.

Meyer, S., Healy, P., Lynn, T., Morrison, J., 2013. Quality assurance for open source software configuration management. In: 2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). IEEE, pp. 454–461. *[PS34].

Middleton, P., Joyce, D., 2012. Lean software management: BBC worldwide case study. IEEE Trans. Eng. Manage. 59 (1), 20–32. *[PS35].

Nagy, A., Njima, M., Mkrtchyan, L., 2010. A Bayesian based method for agile software development release planning and project health monitoring. In: 2010 2nd International Conference on Intelligent Networking and Collaborative Systems (INCOS). IEEE, pp. 192–199. *[PS36].

Neely, S., Stolt, S., 2013. Continuous delivery? easy! just change everything (well, maybe it is not that easy). In: Agile Conference (AGILE), 2013. IEEE, pp. 121–128. *[PS37].

Nilsson, A., Bosch, J., Berger, C., 2014. Visualizing testing activities to support continuous integration: a multiple case study. In: Agile Processes in Software Engineering and Extreme Programming. Springer, pp. 171–186. *[PS38].

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

Olsson, H.H., Alahyari, H., Bosch, J., 2012. Climbing the "stairway to heaven"–a multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA). IEEE, pp. 392–399. *[PS39].
Olsson, H.H., Bosch, J., Alahyari, H., 2013. Towards R&D as innovation experiment systems: a framework for moving beyond agile software development. In: IASTED Multiconferences–Proceedings of the IASTED International Conference on Software Engineering, SE 2013, pp. 798–805. *[PS40].
Orso, A., Liang, D., Harrold, M.J., Lipton, R., 2002. Gamma system: continuous evolution of software after deployment. In: Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 65–69. *[PS41].
Papatheocharous, E., Belk, M., Nyfjord, J., Germanakos, P., Samaras, G., 2014. Personalised continuous software engineering. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. ACM, pp. 57–62. *[PS42].
Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P., 2014. Software development in startup companies: a systematic mapping study. Inf. Software Technol. 56 (10), 1200–1218.
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering. British Computer Society, Swinton, UK, pp. 68–77.
Poppendieck, M., Cusumano, M.A., 2012. Lean software development: a tutorial. IEEE Software 29 (5), 26–32. *[PS43].
Ries, E., 2011. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Random House LLC.
Rodríguez, P., Markkula, J., Oivo, M., Turula, K., 2012. Survey on agile and lean usage in Finnish software industry. In: Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement. ACM, pp. 139–148.
Scacchi, W., Alspaugh, T.A., 2013. Processes in securing open architecture software systems. In: Proceedings of the 2013 International Conference on Software and System Process. ACM, pp. 126–135. *[PS44]
Schwaber, K., 2004, vol. 7. Agile Project Management with Scrum. Microsoft Press, Redmond.
Shaw, M., 2003. Writing good software engineering research papers: minitutorial. In: Proceedings of the 25th International Conference on Software Engineering. IEEE Computer Society, pp. 726–736.
Ståhl, D., Bosch, J., 2014. Modeling continuous integration practice differences in industry software development. J. Syst. Software 87, 48–59.
Staron, M., Meding, W., Palm, K., 2012. Release readiness indicator for mature agile and lean software development projects. In: Agile Processes in Software Engineering and Extreme Programming. Springer, pp. 93–107. *[PS45].
Trimble, J., Webster, C., 2012. Agile development methods for space operations. In: The 12th International Conference on Space Operations. *[PS46]
Trimble, J., Webster, C., 2013. From traditional, to lean, to agile development: finding the optimal software engineering cycle. In: 2013 46th Hawaii International Conference on System Sciences (HICSS). IEEE, pp. 4826–4833. *[PS47]
Turhan, B., Layman, L., Diep, M., Erdogmus, H., Shull, F., 2010. How effective is test-driven development. Making Software: What Really Works, and Why We Believe It. O'Reilly Media, Inc., pp. 207–217.
Unterkalmsteiner, M., Gorschek, T., Islam, A.M., Cheng, C.K., Permadi, R.B., Feldt, R., 2012. Evaluation and measurement of software process improvement a systematic literature review. IEEE Trans. Software Eng. 38 (2), 398–424.
Van Der Storm, T., 2005. Continuous release and upgrade of component-based software. In: Proceedings of the 12th International Workshop on Software Configuration Management. ACM, pp. 43–57. *[PS48].
Vitalari, N., Shaughnessy, H., 2012. The Elastic Enterprise: The New Manifesto for Business Revolution. Olivet Publishing.
Wicenec, A., Parsons, R., Kitaeff, S., Vinsen, K., Wu, C., Nelson, P., Reed, D., 2012. Distributed agile software development for the SKA. In: SPIE Astronomical Telescopes+ Instrumentation. International Society for Optics and Photonics, p. 845106. *[PS49]
Zade, H., Choppella, V., 2012. Functionality or user interface: which is easier to learn when changed? In: 2012 4th International Conference on Intelligent Human Computer Interaction (IHCI). IEEE, pp. 1–6. *[PS50].
Zhang, H., Ali Babar, M., 2013. Systematic reviews in software engineering: an empirical investigation. Inf. Software Technol. 55 (7), 1341–1354.

Pilar Rodríguez (PhD) is a postdoctoral researcher at the Department of Information Processing Science, University of Oulu, Finland. She earned her PhD in Computer Science from University of Oulu and her MSc in Information Technologies from Technical University of Madrid. Before taking her current position, she was a research assistant at Technical University of Madrid. She has worked in several research projects related to software engineering including EU-ITEA2 FLEXI, Cloud Software Program, N4S and VALUE (Finland). Her research interests include empirical software engineering, agile, lean, value creation in the software domain and cognitive biases in software engineering.

Alireza Haghighatkhah is a PhD student and a researcher at the Department of Information Processing Science at the University of Oulu, Finland. He received BSc in Applied Computing from Southern Cross University; Australia in 2010 and MSc in Information Processing Science from University of Oulu, Finland in 2014. He has four years industrial experience in J2EE platform and enterprise software development. His research interests include Evidence-based Software Engineering, Software Process Assessment and Improvement as well as Lean and Agile Software Development. He is currently working in DIGILE Need for Speed research program focusing on continuous deployment and experimentation.

Lucy Ellen Lwakatare is a doctoral student and a researcher of the M-Group at the Department of Information Processing Science, University of Oulu. Ms. Lwakatare received her Master's degree in Information Processing Science at University of Oulu in 2013. Her research interests include: portfolio management, agile software development, release processes and DevOps. She is currently a researcher of the Finnish funded, Need for Speed (N4S) project focusing on processes, models or new ways of working for that enable Finnish software intensive organizations to continuously deliver value to customer in real time.

Susanna Teppola (MSc) has worked as a Research Scientist at VTT Technical Research Centre of Finland since 2000. Susanna has over fifteen years' experience in ICT, her current research interests laying in the area of continuous software engineering, software product/service management and variability. In these areas Susanna has conducted and participated in many industrial and industry-driven research projects and project preparations both at national and international level.

Tanja Suomalainen (MSc) Tanja Suomalainen has received her MSc in Information Processing Science from the University of Oulu in 2006. She has worked at VTT (VTT Technical Research Centre of Finland Ltd.) since 2005. She started her career first as a research trainee and then after graduation as a Research Scientist. Currently she is also a PhD student in the Faculty of Information Processing Science at University of Oulu, Finland. Her research interests include planning in software development, product roadmapping, requirements management, global software development engineering, and agile and lean software development.

Juho Eskeli (MSc) Juho Eskeli received his MSc in Electrical and Information Engineering from the University of Oulu in 2009. He has worked at VTT since 2006 and is currently working as research scientist in Smart lighting and integration concepts team in Oulu, Finland. His research interests include software development, software development tools, internet of things, and embedded systems.

Teemu Karvonen (MSc) works as a researcher at the Department of Information Processing Science at University of Oulu. Teemu's research interests are agile and lean practices and their implications to ICT companies. Currently his research is focused on research themes of DIGILE's research program 'Need 4 Speed'. Teemu has also developed software for Nokia's core network servers, WCDMA base stations and smartphones in total over 10 years.

Pasi Kuvaja (PhD) is Senior Programme Manager and co-founder of international research group M-Group in University of Oulu. His research interests are in Software Engineering including: agile and lean software development, software quality, software process, software process assessment and improvement, embedded systems, and software measurement. He is one of the developers of the BOOTSTRAP methodology and product quality-driven software process improvement approach – PROFES. He has been actively contributing the SPICE project and was co-editor of ISO/IEC 15504 Part 7 – "Process Improvement Guide". He has served as International Programme and Organising Committees of many IEEE, IFIP, IFAC and PROFES Conferences.

June Verner has been involved with software engineering research and teaching in New Zealand, Australia, Hong Kong the United States, United Kingdom, Ireland and Finland for over 30 years. Dr. Verner was a Marie Curie fellow at Keele University 2011–2013, and has recently been a visitor at Oulu University, Finland. She is a conjoint professor of Software Engineering at the University of NSW and a Research fellow at Keele University, United Kingdom.

Markku Oivo (PhD, eMBA) is Head of department and Director of M-Group at University of Oulu, Finland. During 2000-2002 he was Vice President and director at Solid Co. He held several positions at VTT 1986-2000. He had visiting positions at University of Maryland (1990-91), Schlumberger Ltd. (Paris 1994-95), Fraunhofer IESE (1999-2000), University of Bolzano (2014-2015) and Technical University of Madrid (2015). He worked at Kone Co. (1982-86). He has initiated and managed 100+ research projects with tens of millions of euros funding. He has over 100+ publications. His research interests include empirical software engineering, agile, lean, quality & process improvement, and startups.

Please cite this article as: P. Rodríguez et al., Continuous deployment of software intensive products and services: A systematic mapping study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2015.12.015

Series title and number

VTT Science 132

Title	**Changing the planning for agile and lean software development** From roadmapping to continuous planning
Author(s)	Tanja Suomalainen
Abstract	It is hard to survive and succeed in today's business environment, and to be able to sense and respond to predictable and unpredictable events. Also, market uncertainties, increased competitiveness and the constant need to shorten development cycles call for more flexible, responsive and adaptive software development practices. Agile and lean software development practices have been presented as a solution to these challenges and to creating a change-tolerant organisation. Despite the wide adoption of agile and lean practices, it is realised that companies are going further in their practices towards continuous deployment. All of these practices, agile, lean, and continuous deployment, change scheduling and planning, among other things. This thesis investigates how the planning has changed in agile and lean software development from roadmap-based planning towards continuous planning. Roadmapping is seen a process of creating and revising future plans. It is used to manage a high-level view and to link aspects of business to software development, as well as to bridge the gap between different levels of planning. In contrast, continuous planning is a process of implementing the planning practices continuously based on a need, instead of the predefined and regular planning occasions. This thesis provides empirical evidence of how large and global software development companies are conducting planning. The empirical data were collected in two ways: firstly, by conducting an initial inquiry consisting of both questionnaire study and semi-structured interviews, and secondly, by conducting a multiple-case study. According to the results, in software development, roadmap-based planning focuses mainly on product roadmapping, as it improves visibility both upwards to business and strategic planning and downwards to team level planning. The results also show that the main levels of continuous planning are: strategic, financial, business, product, and release planning. The main cycles of planning are conducted weekly, quarterly or annually. Longer-term plans are created for the next three-year period. On the basis of the findings, it was realised that planning practices have changed both in regard to scope and schedule. Planning in agile and lean software development is not restricted to release planning only; instead it is viewed from a wider perspective that involves also strategic and financial planning. What is more, the time frame of the plans has shortened remarkably, from years to months, weeks and days. The reasons for these changes are both internal and external. Both the unstable and turbulent business environment and the rapid development of technology and new product development practices as well as shorter product development cycles are drivers for the change in planning.
ISBN, ISSN, URN	ISBN 978-951-38-8446-8 (Soft back ed.) ISBN 978-951-38-8445-1 (URL: http://www.vttresearch.com/impact/publications) ISSN-L 2242-119X ISSN 2242-119X (Print) ISSN 2242-1203 (Online) http://urn.fi/URN:ISBN:978-951-38-8445-1
Date	September 2016
Language	English, Finnish abstract
Pages	108 p. + app. 126 p.
Name of the project	
Commissioned by	
Keywords	Continuous planning, roadmapping, levels of planning, software development, agile-lean organisation, continuous deployment, change
Publisher	VTT Technical Research Centre of Finland Ltd P.O. Box 1000, FI-02044 VTT, Finland, Tel. 020 722 111

Julkaisun sarja ja numero

VTT Science 132

Nimeke	**Suunnittelun muuttuminen ketterässä ja kevyessä ohjelmistokehityksessä** Tiekarttapohjaisesta suunnittelussa kohti jatkuvaa suunnittelua
Tekijä(t)	Tanja Suomalainen
Tiivistelmä	On vaikea menestyä tai ehkä edes selviytyä nykypäivän liiketoimintaympäristössä eli pystyä vastaamaan paitsi ennustettavissa oleviin, myös arvaamattomiin muutoksiin. Markkinoiden epävarmuus, kova kilpailu ja tarve lyhentää kehitysaikaa edellyttävät joustavampia, reagoivampia ja mukautuvampia ohjelmistokehityskäytäntöjä. Ketterät ja kevyet menetelmät on esitetty ratkaisuna näihin vaatimuksiin, mutta on havaittu, että ohjelmistoyritykset ovat menossa kohti jatkuvaa kehitysmallia. Ketterät ja kevyet menetelmät sekä jatkuva kehitysmalli muuttavat suunnittelua. Tässä tutkimuksessa selvitettiin, miten suunnittelu on muuttunut ketterässä ja kevyessä ohjelmistokehityksessä tiekarttapohjaisesta suunnittelusta kohti jatkuvaa suunnittelua. Tiekarttapohjainen suunnittelu on lähestymistapa luoda ja päivittää tulevaisuuden suunnitelmia sekä hallita ja yhdistää yrityksen liiketoimintanäkymä ohjelmistokehitykseen. Sitä käytetään myös parantamaan näkyvyyttä eri tasojen välillä. Jatkuva suunnittelu tarkoittaa suunnittelun toteuttamista tarpeen mukaan nopeissa rinnakkaisissa kierroksissa ennalta määritettyjen vaiheiden sijaan. Tutkimus perustuu empiiriseen aineistoon siitä, kuinka isot kansainväliset ohjelmistoyritykset toteuttavat suunnittelua. Aineisto kerättiin tekemällä ensin alustava tiedonkeruu, joka koostui kyselytutkimuksesta ja teemahaastatteluista, ja toteuttamalla sen jälkeen monitapaustutkimus. Väitöstutkimuksen tuloksena havaittiin, että tiekarttapohjaisen suunnittelun keskiö ohjelmistoyrityksissä on tuotesuunnittelu. Tiekarttapohjainen tuotesuunnittelu parantaa näkyvyyttä sekä strategian ja liiketoiminnan suunnitteluun että ohjelmistotiimitason vaatimusten suunnitteluun. Tutkimustulokset vahvistavat, että jatkuvan suunnittelun päätasot ovat strateginen, talous-, liiketoiminnan, tuote- ja julkaisusuunnittelu. Suunnittelun pääkierrokset toteutetaan viikoittain, neljännesvuosittain ja vuosittain. Pisimmän aikavälin suunnitelmat luodaan yleensä kolmeksi vuodeksi eteenpäin. Tutkimuksen tuloksena huomattiin, että suunnittelukäytännöt ovat muuttuneet laajuuden ja aikataulun suhteen. Suunnittelu ei rajoitu ketterässä ja kevyessä ohjelmistokehityksessä pelkästään julkaisusuunnitteluun, vaan se nähdään laajemmasta näkökulmasta, johon sisältyvät myös strateginen ja taloussuunnittelu. Suunnitelmien aikataulu on lyhentynyt huomattavasti viime vuosien aikana. Suunnittelua ovat muuttaneet epävakaa toimintaympäristö, teknologioiden nopea kehittyminen, uudet tuotekehityskäytännöt ja lyhenevät tuotekehityssyklit.
ISBN, ISSN, URN	ISBN 978-951-38-8446-8 (nid.) ISBN 978-951-38-8445-1 (URL: http://www.vtt.fi/julkaisut) ISSN-L 2242-119X ISSN 2242-119X (Painettu) ISSN 2242-1203 (Verkkojulkaisu) http://urn.fi/URN:ISBN:978-951-38-8445-1
Julkaisuaika	Syyskuu 2016
Kieli	Englanti, suomenkielinen tiivistelmä
Sivumäärä	108 s. + liitt. 126 s.
Projektin nimi	
Rahoittajat	
Avainsanat	Jatkuva suunnittelu, tiekarttapohjainen suunnittelu, suunnittelutasot, ohjelmistokehitys, ketterä ja kevyt organisaatio, jatkuva kehitysmalli, muutos
Julkaisija	Teknologian tutkimuskeskus VTT Oy PL 1000, 02044 VTT, puh. 020 722 111

Changing the planning for agile and lean software development

From roadmapping to continuous planning

Market uncertainties, increased competitiveness and the constant need to shorten development cycles call for more flexible, responsive and adaptive software development and planning practices. Thus, creating a long-term future plan has become challenging for software development companies.

This thesis investigates how planning has changed for agile and lean software development from roadmap-based planning towards continuous planning. This thesis provides empirical evidence of how large and global software development companies are conducting planning. The empirical data were collected by conducting an initial inquiry consisting of both a questionnaire study and semi-structured interviews, and then, by conducting a multiple-case study involving three case companies.

The results show that planning practices have changed both in regard to their scope and schedule. The scope of planning in agile and lean software development is not restricted to release planning only; instead planning should be viewed from a wider perspective involving also strategic, financial, business, and product planning. What is more, the time frame of plans has shortened remarkably from years down to months, weeks and days. The reasons for these changes are both internal and external, which are elaborated in more details in the thesis.

ISBN 978-951-38-8446-8 (Soft back ed.)
ISBN 978-951-38-8445-1 (URL: http://www.vttresearch.com/impact/publications)
ISSN-L 2242-119X
ISSN 2242-119X (Print)
ISSN 2242-1203 (Online)
http://urn.fi/URN:ISBN:978-951-38-8445-1