CONTINUOUS SOFTWARE ENGINEERING IN THE DEVELOPMENT OF SOFTWARE-INTENSIVE PRODUCTS

TOWARDS A REFERENCE MODEL FOR CONTINUOUS SOFTWARE ENGINEERING

Teemu Karvonen
TEEMU KARVONEN

CONTINUOUS SOFTWARE ENGINEERING IN THE DEVELOPMENT OF SOFTWARE-INTENSIVE PRODUCTS
Towards a reference model for continuous software engineering

Academic dissertation to be presented with the assent of the Doctoral Training Committee of Technology and Natural Sciences of the University of Oulu for public defence in Auditorium IT116, Linnanmaa, on 3 November 2017, at 12 noon

UNIVERSITY OF OULU, OULU 2017
Karvonen, Teemu, Continuous software engineering in the development of software-intensive products. Towards a reference model for continuous software engineering
University of Oulu Graduate School; University of Oulu, Faculty of Information Technology and Electrical Engineering
Acta Univ. Oul. A 695, 2017
University of Oulu, P.O. Box 8000, FI-90014 University of Oulu, Finland

Abstract
Continuous software engineering (CSE) has instigated academic debate regarding the rapid, parallel cycles of releasing software and customer experimentation. This approach, originating from Web 2.0 and the software-as-a-service domain, is widely recognised among software-intensive companies today. Earlier studies have indicated some challenges in the use of CSE, especially in the context of business-to-business and product-oriented, embedded systems development. Consequently, research must address more explicit definitions and theoretical models for analysing the prerequisites and organisational capabilities related to the use of CSE.

This dissertation investigates various approaches to conducting empirical evaluations related to CSE. The study aims to improve existing models of CSE and to empirically validate them in the context of software companies. The study also aims to accumulate knowledge regarding the use of CSE, as well as its impacts.

The case study method is applied for the collection and analysis of empirical data. Twenty-seven interviews are conducted at five companies. In addition, a systematic literature review is used to synthesise the empirical research on agile release engineering practices. Design science research is used to portray the model design and the evaluation process of this dissertation.

Three approaches for evaluating CSE are constructed: (1) LESAT for software focuses on enterprise transformation using an organisational self-assessment approach, (2) STH+ extends the “Stairway to Heaven” model and evaluates company practices with respect to evolutionary steps towards continuous experimentation-driven development, and (3) CRUSOE defines 7 key areas and 14 diagnostic questions related to the product-intensive software development ecosystem, strategy, architecture, and organisation, as well as their continuous interdependencies.

This dissertation states the relevance of CSE in the context of product-intensive software development. However, more adaptations are anticipated in practices that involve business and product development stakeholders, as well as company external stakeholders.

Keywords: agile development, continuous delivery, continuous deployment, continuous experimentation, continuous integration, continuous software engineering, lean enterprise
Karvonen, Teemu, Jatkuva ohjelmistotuotanto ohjelmistopainotteisessa tuotkehityksessä. Kohti jatkuvan ohjelmistotuotannon viitemallia
Oulun yliopiston tutkijakoulu; Oulun yliopisto, Tieto- ja sähkötekniikan tiedekunta
Acta Univ. Oul. A 695, 2017
Oulun yliopisto, PL 8000, 90014 Oulun yliopisto

Tiivistelmä

Jatkuva ohjelmistotuotanto on herättänyt keskustelua nopeasta, samanaikaisesta ohjelmistojulkaisemisesta ja asiakaskokeiluista. Toimintatapa on peräisin Web 2.0 ja software-as-a-service yhteydestä, mutta se tunnetaan nykyleisesti ohjelmistoja kehittevissä yrityksissä. Niin ollen on havaittu tarvitsevat määritelmät ja teoreettisia malleja, joilla voidaan analysoida jatkuvan ohjelmistotuotannon käyttöön liittyviä edellyksiä ja organisaatioiden kykykkyyksiä.

Tässä väitöskirjassa tutkitaan malleja, joilla voidaan arvioida jatkuvaa ohjelmistotuotantoon. Tutkimuksessa pyritään parantamaan nykyisiä malleja ja arviointimallia niiden käyttöä ohjelmistoyrityksissä. Lisäksi tutkimuksella pyritään kasvattamaan tietoa jatkuvasta ohjelmistotuotannosta ja sen vaikutuksista.


Tutkimuksessa rakennettiin kolme tapaa jatkuvan ohjelmistotuotannon arvioimista varten: (1) LESAT for Software keskittyy organisaation muutoskyvykkyyden arviointiin käyttäen itsearviointimenetelmää, (2) STH+, laajentaa "Stairway to Heaven" mallia ja arvioi yrityksen käytäntöjä eri evoluutioaskeleiden matkalla kohti kokeilupaainotteista tuotkehitystä, (3) CRUSOE määrittelee seitsemän pääaluetta ja 14 kysymystä liittyen tuotkehityksen ekosysteemiin, strategiaan, arkkitehtuuriin, organisoimintaan sekä näiden välisiin jatkuvuksiin.

Väitöskirja osoittaa jatkuva ohjelmistotuotannon merkityksen ohjelmistokeskityksessä. Nähtävissä kuitenkin on, että useita nykykäytäntöjä on tarvetta muokata. Erityisesti muokkaustarvetta on tuotkehityksen ja liiketoiminnan sidosryhmien ja yrityksen ulkoisiin sidosryhmiin liittyvissä käytännöissä.

Asiasanat: jatkuva integraatio, jatkuva kokeilu, jatkuva ohjelmistotuotanto, jatkuva toimitus, ketterä kehitys, lean yritys
Dedicated to my grandparents Kerttu and Kalevi
Acknowledgements

My first experiences with agile software development date back to 2007 when the development team with which I was working at the time, among many other Nokia R&D teams, started using Scrum, continuous integration, daily builds, and automated daily testing practices. In a large organisation, such as Nokia, transformation towards agile was an enormous effort, with many uncertainties, hopes, fears, set-backs and occasional improvements, and new innovations. Looking back to those days, I now realize that those ambivalent experiences from the industry have been a very valuable motivator and driver for sustaining my curiosity, which has helped me complete this study.

This dissertation is based on research carried out at the Empirical Software Engineering in Software, Systems and Services (M3S) research unit. I am grateful for my supervisors – research unit leader, Professor Markku Oivo, and research project manager, Docent Pasi Kuvaja – who offered me the research topic, “lean software enterprise”, in the Cloud SW project. This has turned out to be a rabbit hole that just seems go deeper and deeper. I am thankful for all of the trust, support, and guidance along this journey. I am particularly thankful for the opportunity to work full-time in Cloud SW and N4S research projects. It has truly been an inspiring environment to collaborate with wonderful people from the industry, other universities, and research organisations.

I wish to thank Professors Tero Päivärinta and Jürgen Münch for examining this thesis and for their valuable feedback. Moreover, I would like to thank Professor Brian Fitzgerald for accepting the opponent role at my doctoral thesis defence.

I want to thank my follow-up group, Professors Veikko Seppänen, Harri Haapasalo and Doctor Kari Liukkunen. Your suggestions during the early phases of the research gave me new perspectives and useful ideas on how to approach “lean thinking” as a research topic.

Not to forget all the wonderful people of the M3S research unit. I want to especially thank Pilar, Tanja, Lucy, Elina, Woubshet, Ali, and Markus. I cannot close this section without special recognition of Tanja’s and Lucy’s work in conducting “Stairway to Heaven” interviews, countless discussions, and meetings where we analysed data, planned papers, presentations, posters, etc. There are no words to express my gratitude. Thank you so much!
I also wish to express my gratitude to all of my co-authors: Kirsi Mikkonen, Jan Bosch, Helena Holmström Olsson, Tanja Suomalainen, and Marko Juntunen. It has been a privilege to work with such brilliant and helpful people.

I am grateful to my family and my friends. I especially want to thank my parents Hilkka and Jaakko. Your lifelong example has taught me a great deal of important skills and attitudes that have been helpful in the finalising of this work. These learnings will carry on in my life, regardless of what the future may bring. Kiitos!

Milton Keynes, 22.8.2017

Teemu Karvonen
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARE</td>
<td>Agile release engineering</td>
</tr>
<tr>
<td>BM</td>
<td>Business management</td>
</tr>
<tr>
<td>CD</td>
<td>Continuous delivery</td>
</tr>
<tr>
<td>CD2</td>
<td>Continuous deployment</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous integration</td>
</tr>
<tr>
<td>CLOUD</td>
<td>Cloud Software research program</td>
</tr>
<tr>
<td>CMM</td>
<td>Capability maturity model</td>
</tr>
<tr>
<td>CRUSOE</td>
<td>Continuous interdependencies in product-focused software engineering framework</td>
</tr>
<tr>
<td>CSE</td>
<td>Continuous software engineering</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure-as-a-Service business model</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and communications technology</td>
</tr>
<tr>
<td>ICSE</td>
<td>International Conference on Software Engineering</td>
</tr>
<tr>
<td>IES</td>
<td>Innovation experiment systems</td>
</tr>
<tr>
<td>IS</td>
<td>Information systems</td>
</tr>
<tr>
<td>IT</td>
<td>Information technology</td>
</tr>
<tr>
<td>LAI</td>
<td>Lean Advancement Initiative program</td>
</tr>
<tr>
<td>LESAT</td>
<td>Lean enterprise self-assessment tool</td>
</tr>
<tr>
<td>MIT</td>
<td>Massachusetts Institute of Technology</td>
</tr>
<tr>
<td>N4S</td>
<td>Need for Speed research program</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform-as-a-Service business model</td>
</tr>
<tr>
<td>R&amp;D</td>
<td>Research and development</td>
</tr>
<tr>
<td>RCoSE</td>
<td>International Workshop on Rapid Continuous Software Engineering</td>
</tr>
<tr>
<td>RR</td>
<td>Rapid release</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software-as-a-Service business model</td>
</tr>
<tr>
<td>SE</td>
<td>Software engineering</td>
</tr>
<tr>
<td>SLR</td>
<td>Systematic literature review</td>
</tr>
<tr>
<td>SMS</td>
<td>Systematic mapping study</td>
</tr>
<tr>
<td>STH</td>
<td>Stairway to Heaven model</td>
</tr>
<tr>
<td>STH+</td>
<td>Extension of the Stairway to Heaven model</td>
</tr>
<tr>
<td>XP</td>
<td>Extreme programming</td>
</tr>
</tbody>
</table>
Original publications

This thesis is based on the following publications, which are referred throughout the text by their Roman numerals:


# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>9</td>
</tr>
<tr>
<td>Tiivistelmä</td>
<td>11</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>13</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>15</td>
</tr>
<tr>
<td>Original publications</td>
<td>17</td>
</tr>
<tr>
<td>Contents</td>
<td>19</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>27</td>
</tr>
<tr>
<td>1.1 Research overview</td>
<td>29</td>
</tr>
<tr>
<td>1.2 Key concepts</td>
<td>30</td>
</tr>
<tr>
<td>1.3 Research objective</td>
<td>32</td>
</tr>
<tr>
<td>1.4 Research questions</td>
<td>33</td>
</tr>
<tr>
<td>1.5 Structure of the dissertation</td>
<td>34</td>
</tr>
<tr>
<td>2 Research background and theoretical lens</td>
<td>35</td>
</tr>
<tr>
<td>2.1 Introduction to continuous software engineering</td>
<td>37</td>
</tr>
<tr>
<td>2.2 Other related research</td>
<td>38</td>
</tr>
<tr>
<td>3 Research design</td>
<td>39</td>
</tr>
<tr>
<td>3.1 Research process and design overview</td>
<td>41</td>
</tr>
<tr>
<td>3.2 Description of applied research methods</td>
<td>42</td>
</tr>
<tr>
<td>4 Findings</td>
<td>43</td>
</tr>
<tr>
<td>4.1 Constructed models for analysing CSE (Papers I–III)</td>
<td>45</td>
</tr>
<tr>
<td>4.2 Empirical findings on continuous software engineering (Papers I–III)</td>
<td>47</td>
</tr>
</tbody>
</table>
4.2.1 Lean transformation assessment practices at Ericsson ............... 68
4.2.2 Interviews at case companies .................................................... 72
4.3 A systematic literature review (Paper IV) ........................................ 78
4.4 Findings summary ........................................................................ 84
  4.4.1 RQ1: What are the key aspects of CSE that a reference model must encompass? ............................................................... 84
  4.4.2 RQ2: How does CSE manifest itself in contemporary software-intensive product development? ................................. 85
  4.4.3 RQ3: What are the impacts of using CSE? .................................. 85
  4.4.4 RQ4: How is CSE investigated in SE studies? ............................ 85
5 Discussion and conclusion 87
  5.1 Summary of the results ................................................................ 87
  5.2 Threats to research validity .......................................................... 90
  5.3 Limitations of the research ............................................................ 92
  5.4 Future research opportunities ....................................................... 93
References 95
Original publications 101
# Introduction

## 1.1 Research overview

In 2017, the Finnish economy is facing unprecedented opportunities related to the research and development of digitalised products and services. Finland is well recognised in the development of successful software-intensive products; however, the consistent creation of new product innovations now necessitates continuous improvement in the ways of working and the adoption of state-of-the-art software (SW) technologies and development practices. Many established companies in different industries are increasingly aware of the opportunities related to *continuous delivery* (Humble & Farley, 2010) and *lean startup* (Ries, 2011) methods and their consistent usage for introducing new innovations to markets.

Research on *Continuous Software Engineering* (CSE) (Tichy, Bosch, Goedicke, & Fitzgerald, 2015) aims to accumulate software engineering knowledge and solutions for rapid and continuous product development. Closely related to the *lean startup* and *lean enterprise* concepts (Humble, Molesky, & O’Reilly, 2015), the key research themes for CSE can be characterised by “*deep customer insight*”, “*real-time value delivery*”, and “*mercury business*” (Järvinen, Huomo, Mikkonen, & Tyrväinen, 2014).

CSE practices (e.g. continuous delivery and continuous experimentation) have largely originated from the Web 2.0 and software-as-a-service (SaaS) development domains. However, it is still unclear how widely these practices can be applied in different software development contexts. Moreover, it is not clear how the transition from conventional development practices to CSE can be done. Consequently, the goal of this dissertation, as part of the Cloud SW (CLOUD)\(^1\) and Need for Speed (N4S)\(^2\) research programs, is to investigate the models for evaluation of the organisational change and practices towards CSE. The industry partners of CLOUD and N4S programs have expressed the need to evaluate their CSE capability and to provide empirical data for supporting the planning and decision-making processes in the transition towards CSE.

Various previous studies on CSE (Claps, Berntsson Svensson, & Aurum, 2015; Leppänen et al., 2015; Lindgren & Münch, 2015; Olsson, Alahyari, & Bosch, 2012;

---

\(^1\) Cloud Software Finland. A Strategic Centre for Science, Technology and Innovation (SHOK) active in 2010–2013

\(^2\) Need for Speed 2014–2017 (www.n4s.fi)
Rissanen & Münch, 2015) have indicated many challenges in the adoption and use of CSE, especially in the context of business-to-business and product-oriented, embedded systems development. For example, Leppänen et al. (Leppänen et al., 2015) found that some customers could be reluctant to deal with more frequent release cycles. Some domains may have constraints related to business and safety-critical aspects of the system. Some managers and developers may not currently have proper tools and confidence in the use of automated testing for continuous deployments. Moreover, these identified challenges in companies often cross over many organisational functions and research disciplines (e.g. technology, business, and social aspects). These identified challenges, just to mention few, clearly address the need for further investigations and development of solutions for mitigating the challenges related to the use of CSE.

Software engineering research addresses the design of theories and models for analysing and solving problems related especially to software development and management of software intensive projects. Models can be used for analysing of different aspects, such as processes, practices, and organisational capabilities. In addition, models can be used for analysing changes of development practices, such as the change from traditional software development towards CSE. This cumulated knowledge, via extensive use of common models, can be further used by researchers and practitioners for more systematic mitigation of problems related to software development.

This dissertation explores the CSE capabilities of the software-intensive industry. Continuous delivery and experimentation (Jan Bosch, 2012a) are in the main scope of the research. Hence, practices for iterative SW project management (Pressman, 2009) and modern release engineering (Adams & McIntosh, 2016) in particular are investigated. The dissertation contributes to the emerging Rapid Continuous Software Engineering (RCoSE) theme (Fitzgerald & Stol, 2017; Tichy et al., 2015) in software engineering (SE). This study comprises three main elements: (1) the design and evaluation of theoretical constructs and models on CSE (Papers I–III); (2) empirical studies in information and communication technology (ICT) companies, aiming for an understanding of how CSE manifests itself in contemporary software development contexts (Papers I–III); and (3) a systematic literature review (SLR) and synthesis of the existing research in the field of modern release engineering practices (Paper IV).

The empirical case studies in this dissertation are conducted mostly in the product-intensive software development context in large or very large ICT companies. A systematic literature review also provides an exhaustive analysis of
the CSE state of the practice and prevalence in the field, as well as a comprehensive answer on the state of the research and impacts of CSE in the software development domain.

CSE builds on the principles of agile software development (Dybå & Dingsøyr, 2008). The CSE research theme relevance has lately been acknowledged widely in the SE research community. However, the state of the art of CSE is considered to be driven by the industry and consultants, and research is currently lagging behind (Dingsøyr & Lassenius, 2016). Consequently, although agile topics have been investigated for over a decade, explicit theories and models for CSE are still relatively scarce, and the existing knowledge on CSE is often ill-structured and fragmented under the agile research agenda. As a concrete action to increase SE domain awareness of the CSE topics, the RCoSE workshop has been organised annually in conjunction with the International Conference on Software Engineering (ICSE) (International Conference on Software Engineering, 2017). There has also been a special issue on RCoSE in a top-level SE journal (Tichy, Bosch, & Goedicke, 2017) (the Journal of Systems and Software). CSE addresses continuous activities within and between business, strategic planning, development, and system operations functions. Hence, this study also touches upon many aspects of business management and information systems disciplines.

This dissertation summarises findings from four original publications (referred to as Papers I–IV). Twenty-seven (27) practitioners were interviewed in five case companies to analyse their organisational capabilities and the prerequisites for and impacts of the use of CSE. The empirical findings were synthesised with those of other primary studies on CSE identified by conducting an SLR.

The research applied a helical (spiral), iterative approach (Figure 1) typical of design science research (DSR) (Hevner & Chatterjee, 2010a). The research involves cycles where knowledge is incrementally increased via the creation and evaluation of research artifacts (Hevner & Chatterjee, 2010a). Case studies were applied for further evaluation of three designed artifacts: LESAT for SW (Paper I), STH+ (Paper II), and CRUSOE (Paper III).

Table 1 summarises the investigated topics, highlights, and related key theories and methods. Table 2 outlines the authors’ contributions to the four original publications (Papers I–IV).
Table 1. Overview of the dissertation research topics, highlights, and related theoretical background.

<table>
<thead>
<tr>
<th>Publication</th>
<th>Research topics</th>
<th>Highlights</th>
<th>Related SE theories and SW development methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper I</td>
<td>Lean software development, Lean enterprise transformation assessment</td>
<td>Adaptation of LESAT (LESAT for SW) and comparison to Ericsson’s corresponding approach to lean assessment; Mapping of lean transformation assessment practices to clauses of ISO/IEC 12207 standard</td>
<td>Lean Enterprise Self-Assessment Tool (LESAT) (Nightingale &amp; Srinivasan, 2011); Poppendieck et al.’s seven principles of lean software development (Poppendieck &amp; Cusumano, 2012); Womack et al.’s five principles of lean manufacturing (Womack &amp; Jones, 2003)</td>
</tr>
<tr>
<td>Paper II</td>
<td>Innovation experiment systems, Continuous deployment, Continuous integration</td>
<td>Extension (STH+) of the Stairway to Heaven model with related practices in evolutionary steps towards Innovation Experiment Systems; Multiple-case study</td>
<td>Stairway to Heaven model (Olsson, Bosch, &amp; Alahyari, 2013); BAPO model (Linden, Van Der, Bosch, Kamsties, Käänsälä, &amp; Obbink, 2004)</td>
</tr>
</tbody>
</table>
Paper III Continuous software engineering

A holistic framework for analysing prerequisites for continuous software engineering (CRUSOE); Case study

Continuous * model (Fitzgerald & Stol, 2017); ESAO model (Jan Bosch & Bosch-Sijtsema, 2014);

Paper IV Continuous integration, Continuous delivery, Continuous deployment, and Rapid release practices

Synthesis of empirical studies and impacts of modern release engineering practices; Consolidated a list of 14 prerequisites for applying continuous deployment strategy

Modern release engineering pipeline (Adams & McIntosh, 2016); Extreme Programming (Beck & Andres, 2004); continuous delivery (Humble & Farley, 2010);

Table 2. Authors' contributions.

<table>
<thead>
<tr>
<th>Publication</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper I</td>
<td>“Adapting the Lean Enterprise Self-Assessment Tool for the Software Development Domain” investigates the Lean Enterprise Self-Assessment Tool (LESAT) for assessing enterprise capabilities for lean transformation. Karvonen was involved in all steps of the study, including research planning, “LESAT for SW” tool adaptation, mapping of assessment practices to the ISO/IEC 12207 standard, and comparison of the tool with Ericsson’s corresponding approach to lean assessment. Karvonen was also the main author of the paper.</td>
</tr>
<tr>
<td>Paper II</td>
<td>“Hitting the Target: Practices for Moving toward Innovation Experiment Systems” presents a multiple-case study conducted in 5 companies, referred to in this dissertation by the pseudonyms “Korte”, “Naava”, “Louhi”, “Holiuva”, and “Latva”. The paper analyses the case companies’ business, architecture, process, and organising practices regarding the transition towards experimentation-driven development. The paper presents an extension to the Stairway to Heaven model, referred to as “STH+” in this dissertation. STH+ characterises key practices for intermediate steps in the transition towards experimentation-driven development. The study reuses the interview guide with the semi-structured questions, which were originally designed by Holmström Olsson, Alahyari, and Bosch (Olsson et al., 2012) with respect to the Stairway to Heaven (STH) model. Karvonen was involved in the research planning, designing of the STH+ model, company interviews, qualitative data analysis and reporting of the results. Karvonen was also the main author of the paper.</td>
</tr>
</tbody>
</table>
Publication | Contribution
---|---
Paper III | “The CRUSOE Framework: A Holistic Approach to Analysing Prerequisites for Continuous Software Engineering” presents a case study analysing the prerequisites for continuous software engineering (CSE). The case company is referred in this dissertation by the pseudonym “Latva”. The CRUSOE framework is constructed and applied in the data analysis. The paper extends a theoretical framework based on the ESAO model (Jan Bosch & Bosch-Sijtsema, 2014). The case study data in Paper II is extended with three additional interviews conducted by Suomalainen. Karvonen was involved in the research planning, designing of the CRUSOE framework, interviews, qualitative data analysis and reporting of the results. Karvonen was also the main author of the paper.

Paper IV | “Systematic Literature Review on the Impacts of Agile Release Engineering Practices” presents a synthesis of empirical studies on the impacts of continuous integration, continuous delivery, continuous deployment and rapid release practices. Karvonen was involved in all steps of the systematic literature review from the identification of the research topic to article search, selection, synthesis, and reporting of the results. Karvonen was also the main author of the paper.

### 1.2 Key concepts

This section aims to clarify the main concepts frequently used in the original publications (Papers I–IV) and in sections 1–5 of this article-based dissertation. Section 2.1 elaborates further on the main concept of this thesis, “continuous software engineering” and closely related concepts to CSE: “lean software development”, “agile release engineering”, and “innovation experiment systems”. They are referred also as “lenses” as they are used in this dissertation for analysing different aspects (dimensions) of the CSE. These lenses are elaborated in subsections 2.1.1–2.1.3. Other frequently-used concepts in this dissertation are:

- **“Prerequisite”** concept refers generally to capabilities, technologies, activities, and practices that are considered to be needed for the successful use of CSE. This concept has been introduced in Paper III to outline the case company’s situation and the changes from traditional development to CSE. It should be noted that the prerequisite concept also incorporates aspects that organisations may not have power to change (e.g. legislation, ecosystem constraints, and domain-specific regulations)

- **‘Capability’** concept generally refers to organisational abilities and skills. This concept has been used in Paper I to characterise capability maturity levels with respect to enterprise transformation assessment. CSE itself can be considered as an organisational capability (Tichy et al., 2017). This
dissertation especially aims to identify organisational capabilities that contribute to the successful use of CSE.

- ‘Practice’ concept generally refers to situated and predefined activities, methodologies, or working patterns. The practice concept is used in Paper II to characterise ways of working that are typically related to steps from traditional development towards CSE. In this dissertation, the practice concept is used for characterising the implementation of CSE principles and the successful use of CSE.

1.3 Research objective

The research objective of this dissertation is twofold. This dissertation seeks to accumulate knowledge on contemporary interpretations and manifestations of CSE. It also aims to design artifacts, i.e. instantiations of different viewpoints on a CSE reference model. The artifacts aim for a comprehensive representation of the key constructs and their relationships in dimensions of CSE. Models are applied to evaluate the organisational capabilities and prerequisites of CSE. Overall, the dissertation aims to contribute to the emergence of a scientific baseline facilitating future investigations and clarification of different aspects related to CSE.

The research is motivated by: a) the contemporary nature of the CSE topic, b) the challenges identified in the use of CSE, and c) the scarcity of empirical studies and models for evaluating CSE in companies. Moreover, a CSE reference model, when properly tested and validated, could significantly help in defining a standardised way of conducting CSE evaluations. The dissertation also aims for broader societal impact by accumulating knowledge about CSE practices and real-life case examples of the adoption and consequences of the use of the CSE in software-intensive product development. The dissertation could also significantly contribute to the evaluation of the CSE utility in software-intensive product development.

The dissertation objectives are closely related to the objectives of the N4S program (Järvinen et al., 2014) theme, “real-time value delivery”. The “real-time value delivery” research theme, as defined by Järvinen et al. (Järvinen et al., 2014), refers to the investigation of organisational capabilities for change from traditional product-based software business to service-based business, where value can be delivered to the customer at a near real time. The research was initiated in the CLOUD program’s work package (sub-project) for investigating the “lean software enterprise” concept. When CLOUD came to closure at the end of 2013, many of
the participating organisations moved to the N4S program. Consequently, research themes in the N4S program are closely related to the lean software development and lean start-up framework (Järvinen et al., 2014). Hence, researchers in the N4S program could use existing empirical findings and research models and tools that were constructed in the CLOUD program.

1.4 Research questions

The research objective is subdivided into four main research questions that can be elaborated with related sub-questions.

The first research question (RQ1) aims to investigate the different aspects related to modelling of CSE in the software-intensive product development context. Existing SE models and empirical data are applied in the design and evaluation cycles for modelling CSE.

\[ RQ1: \text{What are the key aspects of CSE that a reference model must encompass?} \]

\[ \text{For example, what are the requirements for modelling CSE?} \]

The second research question (RQ2) addresses explorative research of CSE usage in companies and empirical confirmation of the challenges and benefits claimed by earlier studies (Claps et al., 2015; Leppänen et al., 2015; Lindgren & Münch, 2015; Olsson et al., 2012; Rissanen & Münch, 2015). The questions are answered by conducting case studies in companies (Papers I–III).

\[ RQ2: \text{How does CSE manifest itself in contemporary software-intensive product development?} \]

\[ \text{For example, how do practitioners view CSE? What is the prevalence of CSE?} \]

The third research question (RQ3) addresses an analysis of the impacts of using CSE. The research question on CSE impacts aim for critical evaluation and synthesis of the empirical knowledge of the benefits and challenges associated with CSE. This question is answered by conducting an SLR (Paper IV) and also with empirical case studies in companies (Papers I–III).

\[ RQ3: \text{What are the impacts of using CSE?} \]
For example, what are the impacts on software project success? What company functions and software development practices are affected by the CSE? Which stakeholders are involved in and affected by the CSE? What are the key prerequisites in the successful use of CSE in software-intensive projects?

Finally, the fourth research question (RQ4) addresses a systematic analysis of a body of knowledge (academic data bases) on CSE. Moreover, this research question aims at understanding the state of CSE research, trends, and patterns, and at identification of possible gaps in the CSE research. This question is answered by conducting an SLR (Paper IV).

**RQ4: How is CSE investigated in SE literature?**

For example, what research methods are applied in the investigation of CSE? What trends and patterns can be identified in publications related to CSE?

### 1.5 Structure of the dissertation

The structure of this dissertation is as follows. Section 2 provides an introduction of the key concepts of CSE and the theoretical background applied in the original publications. Section 3 outlines the applied research methods and highlights dissertation-related research activities and key milestones. Section 4 summarises the findings of the research, which includes a systematic literature review, a presentation of the constructed models (i.e. LESAT for SW, STH+, and CRUSOE) and the empirical findings from the case studies. Section 5 summarises the results and discusses possible future implications of the research. Threats to the research validity and limitations of the study are also discussed in section 5.
2 Research background and theoretical lens

This section primes the research topics on CSE and provides an introduction to the related research and concepts for CSE. It summarises related research activities and interpretations on CSE together with related SE theories and key findings from earlier empirical studies and literature reviews. Paper III aimed for a holistic analysis of the dimensions related to CSE. In addition, Papers I, II, and IV focused on three important concepts related to CSE (Figure 2): (1) lean software development, (2) agile release engineering, and (3) innovation experiment systems. These concepts are elaborated upon, and their relationship with the CSE research agenda via the dissertation is established.

Fig. 2. Investigated concepts (lenses) in the original publications (Papers I–IV).

2.1 Introduction to continuous software engineering

According to the editors of the special issue on Rapid and Continuous Software Engineering (Tichy et al., 2017), CSE is defined as follows:

Continuous software engineering refers to the organizational capability to develop, release and learn from software in rapid parallel cycles, typically hours, days or very small numbers of weeks. This includes determining [sic]
new functionality to build, prioritize the most important functionality, evolve and refactor the architecture, develop the functionality, validate it, release it to customers and collect experimental feedback from the customers to inform the next cycle of development (Tichy et al., 2017, p. 173).

Definitions of CSE are still prone to contextual variations and interpretations, which are presented in SE literature. The above citation indicates two essential CSE practices: ‘release software in rapid cycles’ and ‘collect experimental feedback’. These practices are addressed often in conjunction with ‘continuous delivery’ (Humble & Farley, 2010), ‘continuous deployment’ (Olsson et al., 2012), and ‘continuous experimentation’ (Fagerholm, Guinea, Mäenpää, & Münch, 2016).

While rapid continuous releasing and continuous experimentation practices can also exist in isolation, CSE addresses them as elements complementing each other (Olsson & Bosch, 2014). The continuous delivery and experimentation practices are elaborated later in the subsections for ‘agile release engineering’ and ‘innovation experiment systems’. The third topic investigated, ‘lean software development’, is often considered as a useful platform upon which other capabilities of CSE are built. According to Fitzgerald et al. (Fitzgerald & Stol, 2017, p. 177), ‘lean thinking’ is a useful lens to view CSE.

CSE addresses rapid and lightweight practices and their integration into parallel rapid development cycles. Although CSE aims for rapid cycles, the duration of cycles could vary from hours to even a few weeks, depending on the case (Tichy et al., 2017). A loose definition of cycles could affect researchers’ and practitioners’ claims about the usage of CSE. It is also often unclear how to determine capability levels related to integrative activities and cycles between strategy planning, development, and operations. Product development activities can also be temporarily accelerated by skipping some steps in the cycle. In agile terminology, this type of activity is often referred as collecting various kinds of technical debt (Behutiye, Rodríguez, Oivo, & Tosun, 2017). Hence, while this dissertation cannot provide a more comprehensive definition of CSE, it views speed (e.g. release cycle duration), the integration of activities (e.g. synchronised planning and development cycles), and stability (e.g. recurrence of development activities in release cycle) as key indicators of consistent CSE capability.

CSE addresses multidisciplinary aspects of software-intensive product development from product ideation and determination of system requirements to post-deployment activities. Adjacent business activities in particular are identified as closely interrelated with CSE. The research in this dissertation primarily
addresses SW development aspects in CSE; however, business and operations viewpoints are also touched upon in a holistic model for analysing software-intensive product development.

Real-life examples of CSE usage in companies have increased significantly over the past five years (Rodríguez et al., 2017). For example, Olsson et al.’s (Olsson et al., 2013) multiple-case study indicated two case companies applying both continuous deployment and innovation experimentation practices. Rahman et al. (Rahman, Helms, Williams, & Parnin, 2015) identified 19 software companies using continuous deployment practices. For example, Facebook, GitHub, Netflix, and Rally Software are successfully applying such practices. Meanwhile, some case studies have indicated several challenges related to continuous deployment and continuous delivery practices. Leppänen et al.’s (Leppänen et al., 2015) multiple-case study findings indicated that companies must often balance continuous deployment frequency with their customers’ needs and capability to receive new releases. Similar kinds of obstacles to CSE have been identified in Atlassian (Claps et al., 2015) and B2B (Rissanen & Münch, 2015) and the telecommunications industry (Debbiche, Dienér, & Svensson, 2014) contexts.

Facebook (Adams et al., 2015; Facebook Developers, 2014) can be considered a prime example of a company applying several elements of CSE in their release engineering; typically, new software is shipped to production twice a day (depending on the platform). Post-deployment end-user behaviour is monitored carefully via multiple end-user metrics and indicators of service usage patterns. In the case of any negative (unsatisfactory) results, such as a dropping in usage of a service or any abnormal behaviour of a system, individual features can be disabled (switched off) from the production release or, if needed, the whole release can be cancelled and rolled back to a previous version within a few seconds.

Several organisational and procedural changes have preceded the usage of CSE. Facebook has invested extensively in systems for the continuous integration (CI) and automation of a deployment process. Each developer committed to a trunk undergoes extensive automated tests in CI systems. Deployments to production are put into a one-day stabilisation period and release branch, which is used to distribute the new functionality, first for limited groups of end-users (alpha and beta groups) and, then gradually, globally to all users of Facebook. Facebook has also changed its organisational structures and development team responsibilities towards feature teams; that is, the same team is responsible for developing a specific feature for all supported platforms (Adams et al., 2015; Facebook Developers, 2014).
Both Rahman et al.’s (Rahman et al., 2015) and Rodríguez et al.’s (Rodríguez et al., 2017) analysis of the continuous deployment application domain indicated a relationship between CSE and web-application development. In contrast, Mäntylä et al.’s (Mäntylä, Adams, Khomh, Engström, & Petersen, 2014) literature review indicated that rapid releases are used in several different software domains and that they are highly relevant to many software development organizations. Many leading edge companies (e.g. Google, Mozilla, Atlassian (Claps et al., 2015)), which are developing both mobile and desktop applications, have recently changed their release strategies, moving towards faster release cycles. The Chrome and Firefox web browsers have both changed from a 12–18 month to a 6-week release cycle. This trend could indicate that, while CSE practices are still perhaps most often applied in website development, they may already increasingly affect and shape practices in many other software development subdomains.

Bosch (J Bosch, 2014) states that modern technology and business enablers (e.g. Web 2.0. SaaS, PaaS, IaaS) are the key drivers towards the use of CSE. Hence, companies can evolutionarily adjust their software-intensive product development practices towards continuous innovation and delivery cycles.

CSE resembles software prototyping techniques (throwaway prototype, evolutionary prototype) for determining system requirements, as described by Pressman (Pressman, 2009). However, in contrast to traditional prototyping, the speed of development cycles is clearly of great interest in CSE. Bosch (J Bosch, 2014) rationalises the transition towards CSE by a faster response to changes in the market and a more accurate alignment to customer needs. Continuous evolutionary service improvements and new product innovation are also seen as a lucrative long-term strategy in many companies (Järvinen et al., 2014).

CSE is one of the emerging themes in SE research (Dingsøyr & Lassenius, 2016). Fitzgerald et al.’s (Fitzgerald & Stol, 2017) roadmap and agenda for CSE identifies several continuous, interlinked activities between business strategy, development, operations, and continuous improvement (Figure 3). The authors therefore suggest that research on CSE must address the understanding of these activities and interrelations and their parallel usage scenarios. Related work undertaken by the DevOps movement (Lucy Ellen Lwakatare, Kuvaja, & Oivo, 2015) addresses operations viewpoints and their relationship with continuous development activities. Fitzgerald et al. argue that a corresponding BizDev link is needed between business strategy and development activities. This point of view also addresses mechanisms for an understanding of how individual activities are related to planning, budgeting, deployment, and release cycles. Related CSE
practices, such as continuous planning (Suomalainen, 2015), continuous integration (Eck, Uebernickel, & Brenner, 2014; Ståhl & Bosch, 2014b), continuous delivery (e.g. (Ries, 2011), (Humble & Farley, 2010), (Duvall, Matyas, & Glover, 2007) and (Humble et al., 2015)), and continuous experimentation (Fagerholm et al., 2016) have already been elaborated upon in several books and SE studies.

The need to theorise (Päivärinta & Smolander, 2015) CSE practices creates various challenges to empirical research. For example, an understanding of the different variations between implementations of CSE practices requires the development and use of rigorous modelling and visualisation techniques, such as continuous integration flows (Ståhl & Bosch, 2014a) and continuous integration visualisation technique (CIViT) (Nilsson, Bosch, & Berger, 2014), for testing activities. Although establishing an understanding of SE practices and their impacts would be highly important, a comprehensive investigation of the impacts of interrelated parallel activities in CSE might turn out to be difficult; hence, in practice, investigations often have to be limited to those within one or a few organisational activities and disciplines related to CSE. Consequently, while some direct impacts can be determined to some extent, other impacts on all software project stakeholders and enterprise levels could be more difficult to anticipate and evaluate. Meanwhile, companies may find it extremely difficult to plan their activities and the key indicators to be monitored in a transition towards CSE;

*Reaching the goal of rapid continuous software engineering is a holistic endeavour; it cannot be addressed only by research in the area of process aspects in software engineering* (Tichy et al., 2015).
Figure 3 illustrates key activities and their relationships to continuous activities in CSE, as defined by Fitzgerald et al. (Fitzgerald & Stol, 2017). The work in this dissertation touches upon various activities and aspects of the CSE research agenda that are outlined in the Continuous * (Star) model (Fitzgerald & Stol, 2017):

- **Paper I** focuses on “continuous improvement” activities and underlying principles related to “lean thinking” in CSE, e.g. flow, continuous improvement, leadership processes, and key practices for determining capabilities for lean enterprise transformation.
- **Paper II** focuses on “continuous development” and “continuous improvement and innovation” activities in CSE. The paper elaborates on the evolutionary steps from traditional innovation and prototyping processes towards innovation experiment systems (Jan Bosch, 2012a). In addition, Paper II defines practices that are typically applied in each evolutionary step (Olsson et al., 2013) towards “continuous experimentation”, including “continuous integration” and “continuous deployment” steps.
Paper III focuses particularly on “continuous business strategy” and “continuous planning” and “continuous development” activities in CSE. It aims to clarify continuous interrelationships and activities between business strategy planning and development. It also elaborates “BizDev” and “DevOps” activities, which are needed to eliminate discontinuities between business strategy and planning, development, and operations.

Paper IV focuses on “continuous development” activities in CSE. The paper synthesises empirical studies on continuous integration, continuous delivery, continuous deployment, and rapid release practices. “Agile Release Engineering” is used as a blanket term in Paper IV to outline the above-mentioned practices associated with the modern release engineering pipeline (Adams & McIntosh, 2016).

Consequently, this dissertation addresses various aspects and concepts related to CSE. However, as illustrated in Figure 3 (Continuous * model), there are many related activities, such as “continuous security” and “continuous run-time monitoring”, that are embedded in CSE concept, but are not investigated in the original publications of this dissertation. The following subsections continue the presentation of three key concepts in CSE that are investigated in this dissertation.

**2.1.1 Lean software development**

MIT’s researchers in the 1980s noticed that Toyota was clearly outperforming its US competitors in terms of efficiency.

*(Toyota) was producing automobiles with roughly half the number of labour hours as the Big Three US automakers (Poppendieck & Cusumano, 2012, p. 26).*

According to Poppendieck and Cusumano (Poppendieck & Cusumano, 2012), the “lean” concept was first associated solely with car manufacturing and management philosophies in the Toyota Motor Company.

Womack and Jones (Womack & Jones, 2003) argued that lean principles can also be applied outside of the car manufacturing context. They summarised *lean thinking* as follows. Lean thinking emphasises a comprehensive understanding of customer value and waste in production. Waste refers to anything that does not add value for the customer. In addition, all value streams from raw materials to
delivering a product to a customer should be comprehensively mapped and evaluated. Once all steps contributing to value and waste are understood, the focus should be moved to a continuous process improvement of waste-elimination activities (Womack & Jones, 2003). These improvement activities should be embedded systematically in production cycles, e.g. processes, tools, and techniques. Lean thinking (Womack & Jones, 2003) emphasises the increased understanding of an enterprise’s value stream and the continuous flow of value within and between stakeholders involved in a value stream.

Lean principles have been applied in many industry domains, including the development of software-intensive products and services. The term “lean” became more widely known later in the 1990s in conjunction with just-in-time production and continuous improvement (Kaizen) management philosophies. Interestingly, although Toyota’s production principles were comprehensively investigated, other car manufacturing companies’ attempts to replicate Toyota’s way of working have turned out to be challenging (Womack & Jones, 2003). The main challenge is often related to changing the working culture and mindset with respect to how people perceive their own roles and responsibilities in a company.

In recent years, several authors in the SE domain have addressed lean principles in conjunction with agile development practices in individual developer- and team-level activities (Dybå & Dingsøyr, 2008) and the scaling of agile to the enterprise level (Humble et al., 2015). In particular, Mary and Tom Poppendieck’s (Poppendieck & Poppendieck, 2007) work on adapting lean principles to software development has resulted in several definitions for principles and practices of lean software development. In particular, more recently, lean terminology has been expanded and used in conjunction with methods that only share a very few recognisable characteristics with the original lean manufacturing principles identified in Toyota. Books on lean enterprise (Humble et al., 2015) and lean startup (Ries, 2011) characterise the practices that leverage modern information technology and experimentation-driven flexible business approaches.

To summarise, this dissertation views lean principles and practices as a useful and industry-proven baseline for applying CSE in software-intensive product development. Paper I investigates a lean enterprise transformation assessment tool designed originally for the aerospace and military industry. Paper III extends Fitzgerald et al.’s research on interdependencies related to CSE and their notion of lean as a useful and relevant lens for CSE. Related concepts, such as BizDev, DevOps, and continuous experimentation and innovation (Fitzgerald & Stol, 2017) practices, can be thus linked to a legacy of engineering principles that have already
existed and evolved for several decades, even before software was first integrated into any products or services.

**Lean Enterprise Self-Assessment Tool (LESAT)**

The dissertation addresses lean aspects in CSE via analysing a tool for lean transformation assessment. In conjunction with the CLOUD program objective for lean software enterprise assessment, LESAT (Nightingale & Mize, 2002) was identified as a substantial approach to investigating the lean enterprise concept in the software development domain. The study also explored the adaptability of lean practices from the aerospace industry to software development.

LESAT is based on the Lean Advancement Initiative (LAI) work involving lean principles in aerospace systems acquisition, development, and production processes (Nightingale & Mize, 2002). The work in Paper I extends LAI’s research by adapting LESAT assessment for the software development context. LESAT’s assessment practices were analysed, adapted, and mapped to the ISO/IEC 12207 (ISO, 2008) standard (software lifecycle processes) and were then compared to Ericsson’s corresponding lean assessment survey approach, called a lean amplifier. Rodriguez et al. (Rodríguez, Mikkonen, Kuvaja, Oivo, & Garbajosa, 2013) later investigated lean amplifier usage in Ericsson’s R&D teams.

LESAT’s design is based on a Capability Maturity model (CMM) approach to assessment, i.e. maturity matrices are implemented together with statements that define five capability levels for lean enterprise transformation. LESAT defines lean practices and practices that characterise company behaviour through five capability levels, with level five representing world-class capability in terms of lean behaviour. Each lean practice contains statements and examples that characterise lean transformation capabilities at different maturity levels. In this dissertation, the LAI research on LESAT has been extended by applying adaptations of lean principles to software development. Paper I applies Womack et al.’s (Womack & Jones, 2003) principles of lean thinking, and Mary and Tom Poppendieck’s (Poppendieck & Cusumano, 2012) principles of lean software development.

**2.1.2 Agile release engineering**

CSE addresses rapid and automated practices for the build, integration, and release of software. Consequently, in Paper IV, these development practices are explicitly
analysed in conjunction from a modern release engineering viewpoint. Adams et al. describe release engineering as follows:

Release engineering is the process responsible for taking the individual code contributions of developers and bringing those to the end user in the form of a high quality software release (Adams & Mcintosh, 2016, p. 78).

Release engineering and its related practices have been identified as a salient research area for the CSE (Adams & Mcintosh, 2016; Fitzgerald & Stol, 2017). In this dissertation, the practices investigated are:

- Continuous integration
- Continuous delivery
- Continuous deployment
- Rapid release

The agile release engineering (ARE) concept was introduced and used as a blanket term in the SLR presented in Paper IV. ARE consists of four closely related and complementary practices associated with CSE: rapid release, continuous integration, continuous delivery, and continuous deployment. While these four concepts are often used interchangeably in SE literature, modern release engineering views these practices as also having differences, especially regarding the release decision making, levels of automation, and modes of operation describing how a release engineering pipeline is implemented (Adams & Mcintosh, 2016). CI practice, as described by Beck (Beck, 1999), aims to frequently integrate new code with the system:

When integrating, the system is built from scratch and all tests must pass or the changes are discarded (Beck, 1999, p. 71).

Continuous integration, delivery, and deployment, i.e. the different modes of operation in release engineering, can be described as follows:

- **Continuous integration** practice involves the automated building of binaries and the execution of basic-level testing, but it does not necessarily involve deployment to production and release steps.
- **Continuous delivery** practice involves the deployment pipeline (automated mechanism) to a production system but does not necessarily involve an automated release to the end-user. That is, the release decision can be made separately (and manually).
Continuous deployment practice involves the whole automated release pipeline production, and all the individual code changes are automatically released to production if tests are passed.

The above definitions simplify the reality of release engineering in companies. Interpretations and implementations of continuous integration have been shown to vary strongly from case to case (Ståhl & Bosch, 2014a). Consequently, while continuous deployment and rapid release concepts are often used in conjunction with case studies to describe a certain way of working, in practice, the actual implementation could vary even very radically between cases.

The rapid release concept in SE literature (Mäntylä et al., 2014) is commonly associated with the Mozilla Firefox and Google Chrome web browsers’ release strategies. Firefox uses four code repositories: Mozilla-central, Mozilla-aurora, Mozilla-beta, and Mozilla-release (Mozilla, 2017) for releasing new main versions of the browser every six weeks. The main philosophy in modern release engineering is that manually repeated steps in the release engineering pipeline are minimised. However, different stages of testing, some of them manual, could affect how quickly individual code changes can be released to users. For example, in the case of the Firefox browser release process, a stable release version is built via the testing of a new functionality in the above-mentioned four repositories.

ARE practices have been investigated in various SE studies, especially related to agile methodologies and extreme programming (Beck & Andres, 2004). Rodriguez et al.’s (Rodriguez et al., 2017) systematic mapping study (SMS) on continuous deployment concept indicated the following:

- Fifty relevant primary studies were published on continuous deployment up until 2014.
- Scientific contributions on continuous deployment were considered ‘highly relevant’ from an industry point of view, but their scientific rigour was considered only ‘medium-low’.
- Publications’ yearly distribution trend indicated a significant increase in continuous deployment-related articles during the 2012–2014 period. (The yearly distribution of primary studies is illustrated in Figure 9.)
- The majority of primary studies were industry reports and case studies.
- Ten recurrent main themes were identified among the primary studies. (The themes are elaborated in Table 3.)
Empirical evidence on the benefits and challenges of continuous deployment is limited mainly to practitioners’ perceptions.

A large proportion of industry reports among primary studies in SMS (Rodríguez et al., 2017) supports Dingsøyr et al.’s (Dingsøyr & Lassenius, 2016) notion of CSE as an industry-driven rather than a research-driven phenomenon. As Dingsøyr et al. describes it,

*The state of the art is driven by industry and consultants, and research is lagging behind in synthesizing and systematizing knowledge and helping to validate or dismiss the many claims made by proponents for various tools and techniques* (Dingsøyr & Lassenius, 2016, p. 58).

Recurrent themes related to continuous deployment (Table 3) indicate a close relationship to modern release engineering pipeline activities (e.g. “automation”, “configuration management”, and “fast and frequent release”) and interdependencies with “agile and lean”. In addition, “customer involvement” and “continuous and rapid experimentation” themes clearly refer to the CSE capability of continuous experimentation.

**Table 3. Factors of continuous deployment according to Rodriguez et al. (Rodríguez et al., 2017).**

<table>
<thead>
<tr>
<th>Themes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Fast and frequent release</td>
<td>Ability to release software whenever the organisation wants to (on demand or at will) based on need and with preference given to shorter cycles or even continuous flow (weekly or daily).</td>
</tr>
<tr>
<td>2. Flexible product design and architecture</td>
<td>CD requires evolutionary and robust software architecture with the aim of balancing speed and stability.</td>
</tr>
<tr>
<td>3. Continuous testing and quality assurance</td>
<td>Ensuring the quality of the software at all times without compromises, despite the need for fast and continuous deployment</td>
</tr>
<tr>
<td>4. Automation</td>
<td>Automating the delivery pipeline from building and testing to deployment and monitoring</td>
</tr>
<tr>
<td>5. Configuration management</td>
<td>Version control branching strategies and system configuration management approaches to enable CD</td>
</tr>
<tr>
<td>6. Customer involvement</td>
<td>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</td>
</tr>
</tbody>
</table>
The research in Paper IV (SLR) extends Rodriguez et al.’s mapping study (Rodriguez et al., 2017) on continuous deployment by focusing on a synthesis of empirical studies to investigate how ARE practices have been studied and what their impacts are.

### 2.1.3 Innovation experiment systems

Continuous improvement, innovation, and experimental customer feedback are viewed as essential elements of CSE (Tichy et al., 2017). Customer feedback is used to determine the requirements for the next development cycle and to enable continuous build-measure-learn cycles for improvement and innovation activity (Fitzgerald & Stol, 2017). Moreover, carefully incorporating business and customer stakeholder feedback in development cycles is emphasised in agile development methodologies, such as extreme programming:

*We need to control the development of software by making many small adjustments, not by making a few large adjustments, kind of like driving a car. This means that we will need the feedback to know when we are a little off, we will need many opportunities to make corrections, and we will have to be able to make those corrections at a reasonable cost.* (Beck & Andres, 2004, p. 30)

Customer-related experimentation practices have been addressed especially in the context of web development (Kohavi, Henne, & Sommerfield, 2007). More recently, experimentation practices have been incorporated into a model for continuous experimentation (Fagerholm et al., 2016).

The innovation experiment systems (IES) (Jan Bosch, 2012a) concept addresses R&D’s role in conducting rapid experiments with customers to validate whether a new product idea, new feature, etc., truly meets real customer needs and
expectations. Consequently, R&D is considered as a “system” for validating hypotheses on product features and their business potential. Traditionally, hypotheses on customer needs have been created in isolation from R&D by product managers and have later been translated to user requirements (Pressman, 2009) that were frozen in the early phases of the project. IES emphasises an evolutionary product development approach to determining customer needs by frequently deploying new versions and testing as many ideas with customers as possible (Jan Bosch, 2012a).

Fig. 4. Stairway to Heaven model. Adapted from (Olsson & Bosch, 2014).

The transition from traditional development towards IES could be a challenging and long-term journey. Olsson et al. (Olsson & Bosch, 2014) have outlined a roadmap “Stairway to Heaven” (STH) on how, step by step, companies evolve their product development practices. The STH model, illustrated in Figure 4, describes the evolutionary steps (A–E) from traditional development towards IES.

Olsson et al. describe the evolving of software development practices as follows:

As a summary, companies evolving from traditional development start by experimenting with one or a few agile teams. Once these teams are successful, agile practices are adopted by the R&D organization. As the R&D organization starts showing the benefits of working agile, system integration and verification becomes involved and the company adopts continuous integration. Once continuous integration runs internally, lead customers often express an interest to receive software functionality earlier than through the normal release cycle. They want continuous deployment of software. The final step is where the software development company collects data from its
customers and uses the installed customer base to run frequent feature experiments (Olsson & Bosch, 2014, p. 16).

In related case studies conducted by Olsson et al. (Olsson et al., 2013), they could identify companies at different stages (steps A–E) of transition towards IES. In this dissertation, the STH model is applied as a main theoretical lens for analysing the case companies’ CSE capability. Paper II extends the research on the STH model by elaborating key practices for characterising each step in the model. The research in Paper II replicates the case study research conducted by Olsson et al. (Olsson et al., 2013). The original interview plan used by Olsson et al. was applied in a multiple-case study presented in Paper II. Subsequently, the extension of the STH model (STH+) has also been developed in collaboration with Software Center\(^3\) researchers – Helena Holmström Olsson and Jan Bosch.

2.2 Other related research

This section elaborates upon other theoretical background and SE models that were applied in the dissertation. Many of these theories and models were not originally related to CSE, but they have been considered to be useful approaches for analysing CSE.

The research in Paper I extends the research of the Lean Advancement Initiative (LAI) (Nightingale & Srinivasan, 2011) of MIT. This research track also closely relates to the organisational CMM approach. Organisation capability assessment models are elaborated in section 2.2.1.

The research in Papers II and III also applied BAPO (Linden, Van Der et al., 2004) and ESAO (Jan Bosch & Bosch-Sijtsema, 2014) models in a holistic analysis of practices and prerequisites for CSE. The study further elaborates on Fitzgerald et al.’s (Fitzgerald & Stol, 2017) interpretation of CSE activities and their interdependencies, which are presented in Figure 3. The ESAO and BAPO models are elaborated upon in section 2.2.2.

2.2.1 Approaches to organisational capability assessment

As stated earlier, CSE has been defined as organisational capability (Tichy et al., 2017). Consequently, this dissertation and, especially, Paper I touch upon aspects

\(^3\) Software Center (http://www.software-center.se/)
of capability assessment. This section first outlines conventional approaches to capability assessments in the SE domain. It also outlines how these approaches have influenced the LESAT and STH approaches, which are investigated in this dissertation.

**ISO/IEC 15504 and Capability Maturity Model**

Approaches to software development capability maturity assessment were developed in response to the need to reduce risks related to budget overruns, poor quality, and delays regarding software projects and their outcomes. Thus, in the 1980s, the U.S Department of Defense contracted the Software Engineering Institute\(^4\) (SEI) to develop an approach to organisational process assessments, which allowed the ranking of software contractors by their capabilities. This initiative resulted in a first version of the Capability Maturity model (CMM). CMM and its offspring, CMMI (Software Engineering Institute, 2010), have been used widely in software development organisations globally. CMM also later inspired several related approaches to process improvement and capability assessment in various contexts, such as LESAT, which is investigated in this dissertation.

Nightingale et al. (Nightingale & Mize, 2002) elaborate on how CMM was applied as a baseline for the design of LESAT. The applied measurement scale in CMM and levels related to the maturity dimension can also be associated with the five evolutionary steps towards IES of the STH model (Figure 4).

CMM and other conventional assessment models, such as ISO/IEC 15504 (Loon, 2007), have adopted a process-oriented approach to characterise key activities and implementation steps in software systems development. The ISO/IEC 15504 standard defines ‘process’ as a set of interrelated activities, which transform inputs to outputs (Loon, 2007). The Process Reference model, as defined in the ISO/IEC 12207 standard, provides a “process dimension”, i.e. process descriptions that are applied in conjunction with the assessment model. An organisation’s capabilities are determined by first analysing whether certain processes exist and then by evaluating related activities, such as how rigorously a process is followed.

---

\(^4\) Software Engineering Institute at Carnegie Mellon University (http://www.sei.cmu.edu/)
2.2.2 Approaches to holistic analysis of software-intensive product development

The Continuous * model (Fitzgerald & Stol, 2017) addresses a holistic view of continuous activities from strategic planning to operations and continuous improvement. Therefore, this dissertation also views other holistic approaches to analysing product development. The CSE clearly has effects on multiple dimensions related to product development. In addition, as identified by Olsson et al. (Olsson & Bosch, 2014), CSE can also have many implications at the ecosystem level. This section outlines the main characteristics of the BAPO and ESAO models.

Business-Architecture-Process-Organisation (BAPO)

According to Linden et al. (Linden, Van Der et al., 2004), the four-dimensional BAPO model was developed for software product family evaluation purposes. The dimensions are business, architecture, process, and organisation. These dimensions represent the key concerns associated with software development:

- **Business**: how to make profit from your products
- **Architecture**: technical means to build the software
- **Process**: roles, responsibilities, and relationships within software development
- **Organisation**: actual mapping of roles and responsibilities to organisational structures

In this dissertation, the BAPO model is applied in conjunction with defining the extension of the STH model, i.e. practices that characterise each step in the STH model.

BAPO dimensions are considered to have interdependencies (Linden, Van Der et al., 2004). Consequently, changes in any one dimension can have consequences for the others. This also could have implications related to CSE. Strategic planning and the consequences of interdependencies in CSE are addressed in Paper III. Practices related to the continuous planning (Suomalainen, Kuusela, & Tihinen, 2015) activity in CSE are also discussed.

Ecosystem-Strategy-Architecture-Organising (ESAO)

According to Bosch et al. (Jan Bosch & Bosch-Sijtsema, 2014), the ESAO model extends the BAPO model and provides an evolutionary update for the four BAPO
dimensions. Company internal and ecosystem dimensions are also incorporated with other software development concerns, which were introduced in BAPO. ESAO dimensions are mapped to software development concerns as follows:

- The Strategy dimension relates to business concerns
- The Architecture dimension relates to architecture concerns
- The Organising dimension relates to organisation and process concerns.

Bosch et al.’s (Jan Bosch & Bosch-Sijtsema, 2014) ecosystem perspective extends each of these dimensions by separating company internal- and ecosystem-related concerns. In this dissertation, the ESAO model was applied in the construction of the CRUSOE framework, presented in Paper III.
3 Research design

This section outlines the research methods, research schedule, key milestones, and rationales for the research design. The research described in Papers II and III applies guidelines for a case study method (Runeson & Höst, 2008). Data from case studies were used to validate the designed research artifacts’ utility for the evaluation of the case companies’ capabilities for CSE. An ethnographic interview method with an interpretive research approach is applied in the case studies. That is, a case study aims to understand the phenomena through the participants’ interpretation (Runeson & Höst, 2008).

The research described in Paper IV applies guidelines for a systematic literature review (SLR) (Kitchenham, 2004) method. The main research process and activities related to theoretical model design and evaluation are portrayed using guidelines for the design science research (DSR) (Hevner & Chatterjee, 2010a).

3.1 Research process and design overview

The research goal and plan (Karvonen, Oivo, & Kuvaja, 2015) for this dissertation was presented in the doctoral symposium of the XP2015 conference. The original research plan, created in 2011, has been iteratively modified and elaborated based on feedback from other SE researchers, doctoral research follow-up group, and supervisors of this dissertation. The research goal and activities were inspired by the DSR approach (Hevner & Chatterjee, 2010a). Consequently, in this dissertation, DSR is used for portraying the whole dissertation research process. Papers I–IV in this article-based dissertation can be considered as intermediate reports (snapshots) of different phases of the dissertation research process.

The research process in this dissertation can be considered as helical (spiral) and iterative by nature (Hevner & Chatterjee, 2010a) (See Figure 1). Cyclical research activities in DSR are characterised by three research cycles: the relevance cycle (REC), the design cycle (DEC), and the rigour cycle (RIC). RIC cycles were conducted to comprehensively understand the key factors and dimensions of CSE and to make new additions to the knowledge base, primarily by reporting key findings from empirical case studies. REL cycles were conducted in the application domain (software-intensive industry) to determine the CSE topic relevance and requirements for the CSE reference model. DEV cycles were conducted to design research artifacts and evaluate them by using them to investigate and report ICT companies’ capabilities for the usage of the CSE.
Figure 5 shows a simplified illustration of the three main steps that characterise one iteration (cycle) in research. Each cycle resulted in an artifact (model), each of which was separately presented in Papers I–III. This means that, in this dissertation, a design cycle was repeated thrice. The research in Paper IV included an SLR, which contributed to building knowledge, especially for the design of the CRUSOE framework presented in Paper III. The first step in the research cycle focuses on developing a comprehensive understanding of CSE or of a specific aspect related to CSE (e.g. related theories, challenges in application, existing solutions, etc.). This is considered a key prerequisite for moving to the next step, as it allows identification of existing knowledge in the field. The main research method in this first step is a systematic scanning of SE literature, especially related to agile and lean software development. The first step also allows the identification and adoption of prominent existing approaches, which can be used as a baseline for the research.

Fig. 5. Main steps and sequences in the research.

The second step focuses on designing a model (research artifact) that can be used to analyse one or more aspects (dimensions, practices, etc.) of CSE. The third step focuses on an evaluation of the design in the previous step. Research artifacts are evaluated by conducting case studies in ICT companies. Finally, the empirical findings can be further used to increase the knowledge on CSE and for improvement of the research artifacts. Figure 5 illustrates the sequences of the key steps in the research cycle. As illustrated by arrows, sequences between steps are
often bidirectional; that is, new empirical knowledge and experiences can also affect the design of the research artifacts.

All research activities were conducted at the University of Oulu, M3S research unit’s projects as part of the CLOUD and N4S programs. Figure 6 illustrates the research highlights and timeline of the original publications. All empirical research data collection and analyses were performed between 2011 and 2016.

Fig. 7. Theory building process. Adapted from (Hevner & Chatterjee, 2010b, p. 34).

Observations, categorisations, and the construction of statements associated with the models presented in this dissertation can be seen as a process of building theory for CSE. Hevner et al. (Hevner & Chatterjee, 2010b) describe the cycle of theory building as a pyramidal, iterative process that ideally starts at the observation stage, where researchers observe, describe, and measure the phenomena. The next stage is classification, i.e. categorization based upon attributes of phenomena. In the third stage, i.e. defining relationships, the researcher should explore the association between the category-defining attributes and outcomes observed. Figure 7 illustrates the dissertation contributions and stages in theory building. The theory-building process is described by Hevner et al. as follows:

Theory building is a time-consuming and laborious effort that often is a result of several researchers in community working together to put different parts of the puzzle (Hevner & Chatterjee, 2010b).

---

5 Empirical Software Engineering in Software, Systems and Services research unit (http://www.oulu.fi/m3s/)

47
DSR, as defined by Hevner et al. (Hevner & Chatterjee, 2010a), aims to solve human problems via the creation of artifacts. According to Simon (Simon, 1997), as referred to by Hevner et al. (Hevner & Chatterjee, 2010a), the term artifact is used to describe something that is artificial, or constructed by humans, as opposed to something that occurs naturally. A purpose of artifacts in DSR is to improve solutions to a problem or provide a first solution to an important problem. According to Hevner et al. (Hevner & Chatterjee, 2010a), artifacts in IT could be defined as follows:

- Constructs (vocabulary and symbols)
- Models (abstractions and representations)
- Methods (algorithms and practices)
- Instantiations (implemented and prototype systems)
- Better design theories

DSR influenced all research activities that were performed in this dissertation. The idea for construction of a CSE reference model was also inspired by DSR’s design-oriented approach to conducting scientific research. The main problem expressed by the N4S industry partners was the need to define an actionable roadmap and evaluation of a company’s capabilities with respect to CSE. In addition, companies were also often keen to understand the key practices and obstacles in moving towards CSE. Consequently, the research in this dissertation aimed for both scientifically accurate and actionable models contributing to accumulated knowledge in both industry and academia.

### 3.2 Description of applied research methods

This dissertation applied multiple methodologies that aimed for use and evaluation of designed models in conjunction with real business cases. Consequently, original publications were reported mostly as case studies based on data collected by interviewing practitioners in real software development contexts.

Paper I applied the constructive research approach for adaptation of LESAT. The main research steps reported in Paper I were as follows: a) conceptual analysis of LESAT, b) LESAT’s adaptation and mapping of assessment practices to ISO/IEC 12207 standard, and c) evaluation of LESAT for SW by comparing it with corresponding Ericsson’s lean assessment approach. Consequently, research in Paper I involved both the design and evaluation aspect of the DSR cycle. Further
empirical research on lean assessments and improvement of LESAT for SW was discontinued for this dissertation in 2012 owing to ending of the CLOUD program.

Majority of original publications in this dissertation (Papers II and III) applied the case study method for empirical data collection, analysis, and reporting. Consequently, the case study method is elaborated more in the following subsection. The selected research approach is observational, including exploratory and descriptive case study elements (Runeson & Höst, 2008). The researcher’s intention is not to change ways of working or software development practices in companies. Throughout the research conducted in the dissertation, the author has considered himself a neutral actor who aims merely to provide knowledge and recommendations regarding CSE, with which the investigated companies may or may not choose to comply. Hence, the research activities aim for an impact mostly in the academic SE research domain. That means increasing academia and industry knowledge regarding CSE, for example with respect to its applicability and limitations in software development. Moreover, the case study method allowed the use of STH+ and CRUSOE in analysis of the empirical data collected from companies.

Finally, Paper IV applied the SLR method that is used for synthesis of empirical knowledge related to CSE. The SLR method is also elaborated later in this section.

### 3.2.1 Case study method (Papers II and III)

The research in Papers II and III applied guidelines for the case study method as defined by Runeson et al. (Runeson & Höst, 2008). In this dissertation, the case study method was applied primarily to understand companies’ software development practices and their capabilities for adopting and using CSE. Case study data were used for the design, evaluation, and improvement of instantiations of CSE reference models. Semi-structured interviews were selected as the primary data collection method. The companies were all participating in the N4S project and, hence, could most easily be involved in the study. Consequently, in terms of quantitative statistical research methodologies, convenience sampling was applied in the selection of cases. However, case studies do not, by their nature, aim for statistical inference.

Case studies in SE research typically aim for empirical understanding of certain phenomena in a particular field and projects where software professionals are working (Runeson & Höst, 2008). Case studies have been classified into various subcategories to specify the researcher’s activities and intentions in solving a
research problem and their relationship to the investigated phenomenon. Case studies can be classified according to different purposes in how the research is conducted. Runeson et al. specify four types of purposes: 1) exploratory, 2) descriptive, 3) explanatory, and 4) improving. The case study purpose in this dissertation was mostly exploratory, i.e. finding out what is happening, seeking new insights, and generating ideas and hypotheses for new research. In addition, case studies in this dissertation can be considered to also have a descriptive purpose for portraying a situation or phenomenon and an explanatory purpose for seeking an explanation of a situation or a problem.

Although case studies are commonly considered useful for investigating contemporary phenomena in their real contexts, the generalisability of individual case study results to other software projects is never warranted. However, the purpose of the case studies should not be considered in terms of generating statistically relevant evidence. Moreover, the purpose of the case studies in this dissertation is to provide a deeper understanding of the contemporary software development practices in companies to be used in the design and evaluation of CSE models.

There are techniques for improving the precision of case study results. Most importantly, precision can be improved by triangulation techniques, e.g. using more than just one data source, observer, methodology, and theory. The case studies in this dissertation used multiple data sources and multiple observers as well as alternative theories and viewpoints to analyse CSE.

The multiple-case study conducted in this dissertation involved five companies in the ICT sector in Finland. Data was collected via interviews conducted in the companies from November 2014 to January 2015. Table 4 lists the industry domains of the companies and the number of interviews and interviewee roles for each case. The selection strategy for interviewees aimed for triangulation, i.e. the involvement of multiple different roles in software development projects. The interview plan included semi-structured questions on four themes:

1. Organisation and current way of working
2. Customer interaction mechanisms/models
3. Strengths and weaknesses in ways of working
4. Benefits and barriers as experienced when moving towards IES.

All interviews were recorded and transcribed. Qualitative data analysis, based on coding of the interviews with predefined themes (e.g. barriers, benefits, and
practices), was performed by three researchers in continuous collaboration. For confidentiality reasons, the case companies involved in the interviews are referred to by the pseudonyms Korte, Naava, Louhi, Hoilua, and Latva.

**Table 4. Case study interviews.**

<table>
<thead>
<tr>
<th>Case pseudonym</th>
<th>Industry domain</th>
<th>Interviewees' roles</th>
<th># of interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Latva</td>
<td>Embedded devices and R&amp;D services</td>
<td>Senior product manager, Software platform product owner, Quality manager, Senior specialist, President of the business segment, Business developer, Scrum Master, Product manager</td>
<td>8 (5+3)†</td>
</tr>
<tr>
<td>B. Korte</td>
<td>Telecommunications</td>
<td>Test automation manager, Senior developer, Program manager, Operations manager, Technical coordinator</td>
<td>5</td>
</tr>
<tr>
<td>C. Naava</td>
<td>Telecommunications</td>
<td>System verification engineer, Program manager, Software architect, Product line manager, Software engineer</td>
<td>5</td>
</tr>
<tr>
<td>D. Louhi</td>
<td>Factory automation</td>
<td>Project manager, Program manager, User experience designer, Product manager, Developer</td>
<td>5</td>
</tr>
<tr>
<td>E. Hoilua</td>
<td>IT services</td>
<td>Product owner, Project manager, Technical service owner, Technical lead</td>
<td>4</td>
</tr>
<tr>
<td>Total: 27 interviews</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.2.2 Systematic literature review method (Paper IV)

SLRs are typically conducted to answer a specific research question by synthesising existing literature on a specific topic (Kitchenham, 2004). The objective of the SLR presented in Paper IV was to update and synthesise the knowledge provided by related studies on CSE. In this dissertation, the SLR focused on release engineering practices, that is, on the development viewpoint in CSE. The related SMS (Rodríguez et al., 2017) provided a tentative analysis of the benefits and challenges of continuous deployment. It was anticipated that the SLR, conducted one year after a comprehensive SMS (Rodríguez et al., 2017) for

---

†Five interviews were conducted by Karvonen, Lwakatare, and Sauvola for Paper II. Three additional interviews were conducted by Suomalainen for Paper III.
continuous deployment, could provide a more detailed analysis of the impacts of CSE. The publication trend for the continuous deployment topic also indicated that there could be a significant amount of new and relevant studies on CSE that were published between the second half of 2014 and the first half of 2015. The SLR also aimed for a more detailed understanding of the quality of empirical research on continuous deployment and on other related modern release engineering practices and the impacts of those practices. SLR was also inspired by the Coat Hanger (Pääväranta & Smolander, 2015) model, which pointed out the importance of understanding the relationships between SW development practices and their impacts.

The research applied Kitchenham’s guidelines (Kitchenham, 2004) for SLRs. Kitchenham points out that the SLR method synthesises existing work in a manner that is fair and seen to be fair. The fairness of the method is related to careful preplanning of the search strategy and detailed reporting of the “hows” and “whys” in each step. These steps aim for a transparency of the research activities, which allows a later evaluation of the author’s research protocol and possible biases in that. Moreover, possible subjective preferences of the author in the selection of studies, e.g. the selection of primary studies that best support his/her research hypothesis, can be more easily identified.

SLRs are referred to as secondary studies because they evaluate and synthesise data from multiple individual studies. Hence, individual studies that contribute to an SLR are called primary studies. The recommended phases and stages of an SLR study are defined by Kitchenham (Kitchenham, 2004) as follows:

1. Planning of the review:
   - Identification of the need for a review
   - Development of a review protocol

2. Conducting the review:
   - Identification of research
   - Selection of primary studies
   - Study quality assessment
   - Data extraction and monitoring
   - Data synthesis.

The final phase in SLR is the reporting of the review. This is a single-stage phase.
The planning of the SLR was started in spring 2015, and the search for primary studies was conducted in June and August of 2015. Primary studies were searched from six databases: Scopus, IEEE Xplore, ACM Digital Library, ISI Web of Science, Science Direct, and Springer. The following search strings were used to identify relevant papers: ((software AND (“continuous deployment” OR “continuous delivery” OR “continuous integration” OR “rapid releas*”)). The search resulted in 619 primary studies. The selection and quality assessment phase limited the number of primary studies to 71. These articles were taken into deeper analysis. The SLR involved two researchers (the author of this dissertation and another PhD student) working in collaboration. The selection and quality assessment steps were duplicated, i.e. both researchers performed paper selection and assessment first individually and then by peer reviewing each other’s decisions. These steps involved a conflict resolution process where the two researchers discussed the selection evaluation criteria and finally could resolve a unanimous decision for each primary study. The key findings of the SLR (Paper IV) are outlined later in section 4.
4 Findings

This section summarises the key findings of the dissertation, which are presented in more detail in the four original publications (Papers I–IV). Research questions in this dissertation were defined as follows:

RQ1: What are the key aspects of CSE that a reference model must encompass?

RQ2: How does CSE manifest itself in contemporary software-intensive product development?

RQ3: What are the impacts of using CSE?

RQ4: How is CSE investigated in SE studies?

4.1 Constructed models for analysing CSE (Papers I–III)

This section summarises findings from Papers I-III and focuses on following research question:

RQ1: What are the key aspects of CSE that a reference model must encompass?

One of the main objectives of the dissertation was related to the modelling of key concepts and their dependencies in the usage of CSE. The investigation focused on a CSE capability evaluation viewpoint and holistic analysis of company practices and interdependencies in CSE.

Related to the DSR approach selected for the research, the study aimed at the improvement of existing solutions for modelling CSE. The baseline approaches (theoretical building blocks) investigated for the design cycles are outlined in section 2. This section outlines the main characteristics of these investigated models, the steps in the design process and key learnings in the design, and the usage of models in analysing qualitative data collected from the five case companies. The learnings are consolidated in this dissertation to answer RQ1. Three artifacts (Hevner & Chatterjee, 2010a) were designed for the analysis of CSE:

1. LESAT for SW. The adaptation of MIT’s LESAT tool focuses on lean software development (i.e. analysing enterprise capabilities and leadership processes in the transformation towards a lean enterprise).
2. STH+. STH+ provides an extension to the STH model and organisations’ evolution towards innovation experiment systems (i.e. analysing enterprise
practices with respect to evolutionary steps in the transition towards innovation experiment systems).

3. **CRUSOE.** CRUSOE provides a holistic approach to analysing continuous software engineering in product-intensive enterprises (i.e. analysing holistic prerequisites for and interdependencies in moving from traditional development to CSE).

<table>
<thead>
<tr>
<th>Table 5. Description of constructed models in Papers I–III.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline approaches</strong></td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>LESAT for SW (Paper I)</td>
</tr>
<tr>
<td>Model description</td>
</tr>
<tr>
<td>Purpose</td>
</tr>
<tr>
<td>CSE dimension</td>
</tr>
</tbody>
</table>

Table 5 outlines the baseline approaches, theories, and different lenses (dimensions) for analysing CSE in Papers I–III.

### 4.1.1 LESAT for SW (Paper I)

The investigation in Paper I addresses the lean software development dimension of CSE. This section outlines the design steps and findings associated with the adaptation of LESAT. The main goal for the adaptation of LESAT was to provide a tool for software-intensive product development that addresses key practices in enterprise transformation in conjunction with lean development principles. Paper I also reports the tentative analysis of LESAT’s applicability to software-intensive product development.

Because LESAT was originally designed for aerospace and military industry purposes (Nightingale & Mize, 2002), the presumptions before the study were that the role of agile and lean software development would have to be emphasised in
the assessment practices and that, in particular, the terminology in LESAT would have to be modified. LESAT represents a process-oriented approach to organisational assessment (Nightingale & Mize, 2002). The CMM approach was applied in the design of LESAT. Nightingale et al. (Nightingale & Mize, 2002) point out that, while CMM focuses on systems and software development processes, LESAT emphasises leadership and integrative management processes. LESAT addresses three process areas and 52 lean practices in assessment:

1. Lean transformation/leadership (28 lean practices)
2. Life cycle processes (16 lean practices)
3. Enabling infrastructure (8 lean practices)

### I.C.2 Enterprise flow

<table>
<thead>
<tr>
<th>CORE LESAT</th>
<th>Lean practice</th>
<th>&quot;Single piece flow&quot; of materials and information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lean indicators</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Examples)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 1</td>
<td></td>
<td>Material and information flows are disjointed and &quot;optimized&quot; process by process. &quot;Push&quot; mentality prevails.</td>
</tr>
<tr>
<td>Level 2</td>
<td>Some primary flow paths have been overhauled to overcome significant barriers to flow.</td>
<td></td>
</tr>
<tr>
<td>Level 3</td>
<td>Primary flow paths are simplified and aligned to the value stream(s), which allows information and material to flow as required.</td>
<td></td>
</tr>
<tr>
<td>Level 4</td>
<td>Material and information flow seamlessly throughout the enterprise.</td>
<td></td>
</tr>
<tr>
<td>Level 5</td>
<td>Material and information flow seamlessly and responsively throughout the extended enterprise.</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 8. LESAT lean practice “I.C.2 Enterprise flow”. 
Fig. 9. LESAT for SW summary sheet.

The adaptation of LESAT included the following main steps: 1) analysis of the key concepts in LESAT, 2) mapping of practices to clauses of the ISO/IEC 12207 standard, and 3) modification of the LESAT practices, indicators, and statements. When the adaptation was done, LESAT for SW was compared to Ericsson’s corresponding lean assessment approach. The modified LESAT documentation, including “The LESAT for Software” (https://goo.gl/9Odqv) tool and LESAT for SW summary sheets (https://goo.gl/Iis5k), have been available for public download via the Internet since 16 April 2012.

LESAT for SW assessments has been conducted in at least three software companies. The results of using LESAT for SW are reported by Cantanhede (Cantanhede, 2014). Section 4.2.4 elaborates on Cantanhede’s key findings on the usage of LESAT for SW. Adaptation in this dissertation covered the lean practices, definitions, terminology, and capability-level descriptions of LESAT. The LESAT assessment process (e.g. the applicability of the LESAT process in companies) was not investigated.

An example of LESAT lean practice, the “I.C.2 Enterprise flow”, is illustrated in Figure 8. An example of the summary sheet (one page) is illustrated in Figure 9. Each lean practice includes the description, indicators, and examples of lean
enterprise practices for different capability levels. In the application of self-assessment using LESAT, both the current (C) capability and desired (D) capability levels are evaluated.

Paper I indicates that LESAT terminology and concepts were considered mostly compatible with the terminology used in the software business and ecosystems-related literature (Jan Bosch, 2012b; Messerschmitt & Szyperski, 2003; Rajala, Rossi, & Tuunainen, 2003). The following key concepts were identified as the most commonly cited terms: a) extended enterprise, b) value stream, c) stakeholder, and d) suppliers (supplier chain, supplier network, and key suppliers). Eighty-seven percent of LESAT practices were considered compatible with the processes and terminology commonly used in software development; seven assessment practices were determined to require modifications for software development. The modifications were affected by the life-cycle processes section owing to terminology referring to the physical characteristics of aerospace production related to logistics and manufacturing. For example, the term “spares” was considered to be alien in the context of immaterial software products. This term was replaced with a description of a proactive maintenance approach related to the lean software development principle, i.e. “build quality in” (Poppendieck & Poppendieck, 2007).

The adaptation of LESAT also included mapping lean practices to the ISO/IEC 12207 standard. Mapping in this case means that a lean practice was considered by analysing the process descriptions of the ISO/IEC 12207 standard. Processes (clauses) considered to be involved in practice were referred to by corresponding lean practices in LESAT for SW. For example, production and manufacturing processes were mapped to software implementation processes, as described in clauses 7.1.1 (Software Implementation Processes) to 7.1.7 (Software Qualification Testing Process). An example of the ISO/IEC 12207 mapping is illustrated in Figure 9 (summary sheet).

4.1.2 STH+ (Paper II)

The STH model (Olsson et al., 2013) characterises the transformation to CSE as an evolutionary process towards continuous innovation and experimentation-driven development practices. Intermediate steps between traditional development and innovation experiment systems (Jan Bosch, 2012a) are characterised by referring to product management, R&D, validation, and customer functions. Figure 4 illustrates the steps in the STH model. Each function related to steps A–B is
characterised by the different ways in which the organisation operates (i.e.
traditional, agile, and short cycles).

Paper II aims to extend the research on the STH model. The STH+ model was
designed in order to further elaborate key practices that characterise each step in
the model (Table 7). In addition, STH+ aimed for assessment of the extent to which
the practice in question has been adopted in a company (Table 6). The BAPO model
(Linden, Van Der et al., 2004), described in section 2, was applied in the
classification of the practices into four dimensions: business, architecture, process,
and organisation. The STH+ model characterises the steps related to these BAPO
dimensions. For example, practices in the Business dimension involve following
steps between traditional development and IES:

1. (Traditional -> Agile R&D): Incorporates the product owner to represent the
customer in the development team.
2. (Agile R&D -> CI): Incorporates the supply chain (component and technology
suppliers) in the development cycle.
3. (CI -> CD): Incorporates lead customers in the development. Renews the
business model, contracts, marketing, and sales strategies.
4. (CD -> IES): Adopts a data-driven, strategic decision-making model.
   Implements A/B testing with the customer.

Table 6. STH+ model levels of adoption (Paper II).

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1: Not adopted (NA)</td>
<td>The practice is not adopted, or it is abandoned.</td>
</tr>
<tr>
<td>Level 2: Team (TE)</td>
<td>The practice is adopted in some teams. Some teams inside the organisation can be ahead of the rest of the organisation.</td>
</tr>
<tr>
<td>Level 3: Product (PR)</td>
<td>The practice is adopted at the product organisation/program level. Some product organisations can be ahead of the rest of the organisation.</td>
</tr>
<tr>
<td>Level 4: Institutionalised (IN)</td>
<td>The practice is fully adopted; it is the standard way of working throughout the entire organisation.</td>
</tr>
</tbody>
</table>
Table 7. STH+ model practices (Paper II).

<table>
<thead>
<tr>
<th>A. Traditional</th>
<th>B. Agile</th>
<th>C. Continuous integration</th>
<th>D. Continuous Deployment</th>
<th>E. Innovation Experiment Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Customer validation at the end of the project</td>
<td>Product owner to represent the customer</td>
<td>Reduce cost of bad quality</td>
<td>Business model transformation</td>
</tr>
<tr>
<td>Architecture</td>
<td>Early on requirement freeze</td>
<td>Lead architect to protect architecture from erosion</td>
<td>Modularisation to improve ability to perform unit &amp; subsystem testing</td>
<td>Componentisation for partial release &amp; rollback mechanism</td>
</tr>
<tr>
<td>Process</td>
<td>Milestone-driven development</td>
<td>Sprints &amp; daily stand-ups</td>
<td>Test-driven development &amp; automated scripts &amp; automated builds</td>
<td>Continuous release process</td>
</tr>
<tr>
<td>Organisation</td>
<td>Large development teams divided into disciplines (testers, architects, etc.)</td>
<td>Feature teams (cross-functional)</td>
<td>V&amp;V function integrated in agile team (continuous integration roles &amp; infrastructure)</td>
<td>UX/system design integrated in team</td>
</tr>
</tbody>
</table>

The descriptions of the adoption levels and practices in STH+ were defined together in workshops arranged with the authors of the STH model (Olsson & Bosch, 2014). Definitions of the STH+ model were applied in the analysis of the case study data collected from the five companies.

4.1.3 CRUSOE (Paper III)

The CRUSOE framework aimed to extend the work presented in Paper II. The main purpose of the CRUSOE framework is to help in the classification of prerequisites for using CSE. The framework (Table 8) is a 2×7 matrix that defines 14 diagnostic questions on how a company addresses continuous activities related to the company’s internal and ecosystem dimensions. The CRUSOE framework design, addressing a holistic viewpoint on product development, was inspired by the Continuous * model (Fitzgerald & Stol, 2017), the ESAO model (Jan Bosch & Bosch-Sijtsema, 2014), and feedback regarding the STH+ model (Paper II).
STH+ model addressed practices in four dimensions; however, it did not explicitly address relationships and possible sequences between dimensions and the adoption of practices. It was also not clear whether, for example, practices related to the business dimension should be established before or after architecture, processes, and organising.

The CRUSOE framework was designed collaboratively in a workshop organised with the authors of Paper III. The purpose of the framework is to facilitate a systematic analysis and structured articulation of company practices and prerequisites for using CSE. Paper III also aimed to clarify the continuous business and strategic planning activities’ relationship to continuous development activities by particularly addressing the business management viewpoint and empirical findings on continuous planning (Suomalainen et al., 2015). Suomalainen’s case study findings (Suomalainen, 2015) indicated multiple variations between timeframes and levels of planning applied in three case companies. While market situations in the companies were clearly pushing them towards shorter planning cycles, none of the investigated companies utilised continuous planning practice throughout their organisation.

![Fig. 10. Simplified illustration of the CRUSOE framework (Paper III, published by permission of Springer).](image)

Figure 10 shows a simplified illustration of the areas addressed in the CRUSOE framework. This framework utilises the dimensions of the ESAO model (Jan Bosch & Bosch-Sijtsema, 2014): (Area 1) Strategy: “ecosystem strategy” and “company internal strategy”; (Area 2) Architecture: “ecosystem architecture” and “company internal architecture”; and (Area 3) Organising: “ecosystem organising” and “company internal organising”.
The CRUSOE framework highlights the interdependencies between the three dimensions. These interdependencies are addressed by Areas 4, 5, 6, and 7 as they overlap with two adjacent dimensions. These areas illustrate integrative activities between the dimensions. Areas 4, 5, and 6 illustrate integrative activities between dimensions, such as the BizDev concept addressed by Fitzgerald et al. (Fitzgerald & Stol, 2017); BizDev activities are primarily associated with Areas 4, 5, and 7. Area 7 illustrates the most overarching, holistic practices for company governance.

Table 8. The CRUSOE framework areas and diagnostic questions (Paper III). Areas 1–3 use definitions of the ESAO model (Jan Bosch & Bosch-Sijtsema, 2014). Published by permission of Springer.

<table>
<thead>
<tr>
<th>CRUSOE Framework Areas 1–7</th>
<th>Analysis Scope: Company Internal</th>
<th>Analysis Scope: Ecosystem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 - Strategy</strong></td>
<td>What are the options for how the company generates revenue now and in the future? (Jan Bosch &amp; Bosch-Sijtsema, 2014)</td>
<td>What are the options that the company has available in its current role in the ecosystem? (Jan Bosch &amp; Bosch-Sijtsema, 2014)</td>
</tr>
<tr>
<td><strong>2 - Architecture</strong></td>
<td>What are the options for technology choices, technical means, and technical structures to build software-intensive products? (Jan Bosch &amp; Bosch-Sijtsema, 2014)</td>
<td>What are the options for how to design interfaces between the company’s internal architecture and related ecosystem partners, such as suppliers providing solutions and firms that build software on top of a product or platform? (Jan Bosch &amp; Bosch-Sijtsema, 2014)</td>
</tr>
<tr>
<td><strong>3 - Organising</strong></td>
<td>What are the options for ways of organising work, ways of working, roles, responsibilities, processes, and tools within software development? (Jan Bosch &amp; Bosch-Sijtsema, 2014)</td>
<td>What are the options for how a company works with customers, suppliers, and ecosystem partners in terms of processes, tools used, ways of working, and ways of organising the collaboration? (Jan Bosch &amp; Bosch-Sijtsema, 2014)</td>
</tr>
<tr>
<td><strong>4 - Strategy &amp; Architecture interdependences</strong></td>
<td>What are the options to connect the internal strategy and architecture? E.g. what are the practices for continuously validating technology choices, technical means, and technical structures that generate revenue now and in the future?</td>
<td>What are the options to connect the ecosystem strategy and architecture? E.g. what are the practices for continuously comparing different ecosystems’ technical capabilities and interfaces that generate revenue now and in the future?</td>
</tr>
<tr>
<td>Areas 1–7</td>
<td>Analysis Scope:</td>
<td>Analysis Scope:</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>CRUSOE Framework</td>
<td>Company Internal</td>
<td>Ecosystem</td>
</tr>
</tbody>
</table>

| 5 - Strategy & Organising interdependencies | What are the options to connect the internal strategy and organising? E.g. practices for continuously adopting efficient ways of organising work, ways of working, roles, responsibilities, processes, and tools. | What are the options to connect the ecosystem strategy and ecosystem organising? E.g. practices for continuously validating investments in ecosystem processes, tools, ways of working, and ways of organising the collaboration in the ecosystem. |

| 6 - Architecture & Organising interdependencies | What are the options to connect the architecture and organising? E.g. practices for continuously refactoring technical structures that provide efficient organising ways of working, roles, responsibilities, processes, and tools. | What are the options to connect the ecosystem architecture and organising? E.g. practices for providing appropriate technical structures for continuous deployments and collaboration with customers and ecosystem partners. |

| 7 - Strategy & Architecture & Organising interdependencies | What are the overarching company governance options for connecting the company strategy with technical architectures and with ways of organising? E.g. practices for enabling a company culture of continuous improvement, experimentation, and innovation. | What are the overarching company governance options for connecting the company strategy with ecosystem interfaces and ways of collaborating with customers and ecosystem partners? E.g. practices for enabling a culture of continuous improvement, experimentation, and innovation with customers and ecosystem partners. |

The areas and definitions of the CRUSOE framework were applied in the analysis of case study data collected from Latva, presented in Paper III.

### 4.1.4 Combined use of models

As stated earlier, this dissertation applied three lenses for analysing CSE. Consequently, the investigated models aim to analyse different, but interrelated, activities in CSE. Figure 3 (Continuous * model) was used as a backdrop for mapping of original publications I–IV to CSE activities. This section aims to analyse how LESAT for SW, STH+, and CRUSOE could be used together and in different phases of the transformation.
The LESAT for SW addresses enterprise-level assessment and, especially, leadership and transformation processes related to organisational capability for transformation and continuous improvement. Moreover, LESAT facilitator guide (Lean Advancement Initiative, 2001) elaborates the process of how and when to conduct the assessment and how to interpret the results. According to facilitators guide, some companies have found it beneficial to conduct the assessment one month prior to their annual business planning activity for greater impact on setting the annual business objectives.

The STH+ addresses organisational transition towards the continuous deployment and experimentation way of working. The STH+ model allows enterprise level evaluation of business, architecture, organisation, and process dimensions in conjunction with the STH model. However, it also allows evaluation and comparison of how individual teams and product programs are applying practices associated with the steps of the STH model.

The CRUSOE framework addresses the ESAO model, i.e., Ecosystem, Strategy, Architecture, and Organisation, dimensions and their continuous interdependencies in conjunction with CSE. Moreover, it aims for the holistic analysis of prerequisites of CSE, which allows identification of the most urgent and the most severe bottle-necks in the use of CSE in the enterprise and ecosystem level.

Although in this dissertation models were applied individually, it is likely that models, if applied in combination, can aid in developing of the most comprehensive understanding of the organisational capabilities with respect to CSE. Figure 11 illustrates an example of how LESAT for SW, STH+, and CRUSOE could be combined and integrated into a one business case using a Plan-Do-Study-Act cycle (Deming cycle) (The W. Edwards Deming Institute, 2017). Investigated models could be used in the plan-phase of the cycle for determination of the current capabilities and organisational strengths and weaknesses with respect to CSE capability.
4.1.5 Evaluation of models

The empirical evaluation of reference models was done by using models for analysing empirical data from case companies. Consequently, the evaluation confirmed that the models, as described in Papers I–III, could be successfully used to analyse qualitative data and to describe companies’ CSE capabilities (e.g. strengths, weaknesses, etc.) with respect to key dimensions of the CSE. The primary data collection method was semi-structured interviews with company employees. The LESAT for SW evaluation was done by comparing LESAT for SW to Ericsson’s corresponding lean assessment approach.

Section 4.2.2 summarises the key findings based on interviews held in five companies: Korte, Naava, Louhi, Hoilua, and Latva. The case study findings presented in this dissertation have been separately reported to company representatives, and interactive feedback sessions were arranged at the conclusion of each case study data analysis. Hence, company representatives confirmed that...
The conclusions based on the interview data were correct before the results were published in Papers II and III.

### Table 9. Company and publication mapping to applied models.

<table>
<thead>
<tr>
<th>Company name/acronym</th>
<th>Paper #</th>
<th>LESAT for SW</th>
<th>STH+</th>
<th>CRUSOE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ericsson</td>
<td>I</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Korte</td>
<td>II</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Naava</td>
<td>II</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Louhi</td>
<td>II</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Holua</td>
<td>II</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latva</td>
<td>II, III</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

The analysis presented in section 4.2.2 was done by first applying the STH+ model in five cases. The analysis indicated that, among the investigated companies, Latva was the most advanced in terms of applying the CSE in product development. Consequently, the CRUSOE framework was used in Paper III for a more detailed investigation of the prerequisites for a transition towards CSE at Latva. Section 4.2.1 also presents the key findings regarding the LESAT for SW comparison to Ericsson’s corresponding lean and agile assessment approach. Table 9 summarises the mapping of the companies with the original publications and models.

Related field testing of the LESAT for SW approach has recently been undertaken as a master’s thesis project by Cantanhede (Cantanhede, 2014). Cantanhede applied LESAT for SW assessments in three Brazilian software companies. The results indicated the applicability of the LESAT for SW in an ICT context. Cantanhede (Cantanhede, 2014) also pointed out that there is a possibility of creating more variations of LESAT for software based on a specific characteristic of the company. There could be a version for larger companies and another for small businesses. Another option would be to modify the content for companies that are currently focused on agile development but would like to start the adoption of lean principles. A third option could be the adaptation of a version for pre-evaluation, which can be used before starting a larger evaluation process. The author of this dissertation provided LESAT for SW documentation, including LESAT assessment guidance, to Cantanhede via email but was not further involved in the actual assessment process, which was conducted independently as Cantanhede’s master’s thesis research.

To summarise key learnings from the evaluations, it became clear that each of the applied models can provide different viewpoints and insights on the evaluation.
of CSE capabilities. Consequently, the ranking of the utility and usefulness of these models is dependent on what aspects are valued most in the evaluation. While all three models aim for a mostly holistic enterprise and ecosystem viewpoint on CSE, they clearly lack granularity, i.e. the details of analysing, for example, a specific business, development, or operations aspect of CSE. Thus, they may be most suitable in situations when an evaluation is made for the first time and when the evaluation aims for a holistic understanding of company capabilities and the identification of company strengths and key areas for improvement.

4.2 Empirical findings on continuous software engineering (Papers I–III)

This section focuses on the following research questions:

RQ2: How does CSE manifest itself in contemporary software-intensive product development?

RQ3: What are the impacts of using CSE?

First, this section outlines Paper I’s findings of comparing the use of LESAT for SW on lean assessment practices to a corresponding approach that has been used in an evaluation of Ericsson’s R&D team’s transition towards lean software development. The comparison indicates key similarities and differences in the interpretation of the lean development and roadmaps for enterprise transformation.

Second, this section summarises Papers II–III, with key empirical findings from 27 interviews conducted in five case companies. The STH+ (Table 7) and CRUSOE framework (Table 8) were used in interpretation of the case companies’ capabilities and prerequisites for using CSE. This section summarises in a cross-case analysis the main barriers and benefits in transition towards CSE as perceived by interviewees.

4.2.1 Lean transformation assessment practices at Ericsson

As a final step in the LESAT adaption to the software development domain, Ericsson’s “lean amplifier” statements for lean assessment were compared with LESAT for SW statements, referred to as lean practices (Nightingale & Mize, 2002). In their related study, Rodriguez et al. (Rodriguez et al., 2013) described Ericsson’s lean amplifier statements’ design process and application in product development.
teams. According to Rodriguez et al. (Rodríguez et al., 2013), Ericsson initiated a change towards lean product development in 2010. Consequently, Ericsson wanted to design a questionnaire-like solution for guiding evaluations related to lean transformation. A lean amplifier consists of statements that specify key roles in the development process and indicators of lean behaviour. For copyright and confidentiality reasons, the lean amplifier cannot be shared publicly in this dissertation. The amplifier design process and the consequences of transformation in Ericsson have been further elaborated by Rodriguez et al. (Rodríguez et al., 2013). Paper I analysed a beta version of the amplifier, with the main difference between the beta and final versions of the amplifier being that the number of statements was reduced from 167 to 114.

LESAT for SW and Ericsson’s amplifier interpretation of lean practices were studied by analysing the statements (lean practices) and the distribution of lean principles applied in the statements. The results of the comparison are illustrated in Figures 11 and 12. The 12 lean principles applied are elaborated upon in Table 10. The study in Paper I applied both Womack et al.’s principles (Womack & Jones, 2003) and Poppendieck et al.’s principles (Poppendieck & Poppendieck, 2007) in the analysis. These principles were considered as a common interpretation of what constitutes lean principles in software-intensive product development.

The analysis was performed by the first author of Paper I, i.e. the author of this dissertation. The results indicated a clearly different interpretation of lean principles between LESAT for SW and Ericsson’s amplifier. Thus, lean practices and their applicability to context and contextual transformation goals would be an important consideration in an evaluation of the usefulness of a lean model. Further evaluations, such as which approach to lean assessment is most accurate and actionable, could not be determined in Paper I.

Results of the comparison indicated several differences between Ericsson’s lean amplifier and LESAT for SW. Most strikingly, LESAT for SW clearly emphasised the leadership role in assessment, while the lean amplifier incorporated individual roles and the team in the main sections of the approach. A “manager” role had the smallest number of statements in the lean amplifier, whereas LESAT practices clearly had the largest number of practices for “transformation/leadership” processes. In comparison of these practices to lean principles, it was found that LESAT for Software practices covered only 8 lean principles, whereas Ericsson’s amplifier statements could be mapped to all 12 lean principles.
Fig. 12. Distribution of practices by main sections in Ericsson’s lean amplifier and LESAT for SW.

<table>
<thead>
<tr>
<th>Lean principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Value</td>
<td>Value is defined as everything for which a customer is willing to pay, and waste refers to everything that absorbs resources but yields no value.</td>
</tr>
<tr>
<td>b) Value stream</td>
<td>The value stream seeks an optimised end-to-end collection of actions for bringing the product to the customer.</td>
</tr>
<tr>
<td>c) Flow</td>
<td>Flow refers to the organisation of activities as a continuous flow by eliminating discontinuities and unnecessary steps (waste) in the value stream.</td>
</tr>
<tr>
<td>d) Pull</td>
<td>Pull implies producing only what is really needed by making the customer’s needs the primary decision driver.</td>
</tr>
<tr>
<td>e) Perfection</td>
<td>Perfection focuses on continuously improving the current implementation of lean principles.</td>
</tr>
<tr>
<td>f) Eliminate waste</td>
<td>Eliminating waste refers to doing only what adds customer value without delays. “Three biggest wastes in software development are: Extra features, churn, crossing boundaries.”</td>
</tr>
</tbody>
</table>
g) Build quality in
Building quality in refers to early defect prevention and identification. “If you routinely find defects in your verification process, your process is defective.”

h) Create knowledge
Creating knowledge refers to amplified learning using frequent feedback loops. “Planning is useful, learning is essential.”

i) Defer commitment
Deferring commitment refers to deciding as late as possible. “Abolish the idea that it is a good idea to start development with a complete specification.”

j) Deliver fast
Delivering fast refers to responding to real customers’ needs. “Lists and queues are buffers between organisations that slow things down.”

k) Respect people
Respecting people refers to providing an expert workforce and delegating responsibility to workers. “Engaged, thinking people provide the most sustainable competitive advantage.”

l) Optimise the whole
Optimising the whole refers to focusing on achieving an overall goal. “Brilliant products emerge from a unique combination of opportunity and technology.”

In conclusion, LESAT for SW was considered to have limitations, especially regarding the evaluation of the application of lean software development practices in an R&D function. However, LESAT for SW’s strengths are clearly related to the evaluation of the entire value stream and leadership activities related to lean transformation. Consequently, it was determined that LESAT for SW may complement lean assessments in the software domain, especially when analysing the leadership, enterprise level and whole value stream aspects in lean transformation.
4.2.2 Interviews at case companies

This section provides a short introduction to the case companies involved in the interviews conducted as part of the dissertation. The section also outlines the case study findings on each company’s CSE capabilities by using STH+ and the CRUSOE framework in the analysis of the interviews.

Case Korte

Korte is a very large ICT company (~100,000 employees globally) developing telecommunications equipment, servers, and related services for cellular and broadband network operators. Software development plays a major role in all products developed by Korte. Modern telecommunications equipment involves highly complex embedded systems. Telecommunications systems are often integrated together with legacy systems (second-, third-, and fourth-generation mobile networks) and multi-vendor network configurations. Consequently, the
deployment of new systems and components to a production network often requires a careful system validation and piloting phases before deployment can be made to a live production environment. Functional and reliable telecommunication networks are also a critical part of modern society and safety. Five Korte employees were interviewed who were developing a compact broadband base station product.

Table 11. STH+ analysis in case Korte (Paper II).

<table>
<thead>
<tr>
<th>Company</th>
<th>Dimension</th>
<th>Traditional</th>
<th>Agile</th>
<th>CI</th>
<th>CD</th>
<th>IES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Korte</td>
<td>Business</td>
<td>NA</td>
<td>IN</td>
<td>TE</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td>NA</td>
<td>IN</td>
<td>TE</td>
<td>PR</td>
<td>TE</td>
</tr>
<tr>
<td></td>
<td>Process</td>
<td>NA</td>
<td>IN</td>
<td>TE</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Organisation</td>
<td>NA</td>
<td>IN</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

The STH+ model was applied in the analysis of the interview data and the determination of Korte’s CSE capability (Table 11). The interviews indicated that Korte had already institutionalised agile practices within and between individual product programs and R&D teams. Practices related to traditional development were considered to be largely abandoned in the company. Continuous integration practices were widely adopted in product development; however, the interviewees pointed out that some teams were still struggling with continuous integration practices. Further, automated testing was also not considered to be at a satisfactory level. Consequently, a continuous deployment capability was considered to be achieved so far only in a few product programs that the interviewees were aware of. Related to organisational structures, company size, and industry stakeholders, the main barrier to continuous deployment was the alignment of internal and external stakeholders to shorter development cycles. In particular, rigorous testing of network systems and certifications were considered problematic in the transition to the continuous deployment mode. In addition, the lack of automated testing, i.e. existing manual steps in testing, was viewed as a hindrance to the transition to continuous deployment. The capability of conducting experimentation with customers was considered to exist only in a few teams related to the architecture dimension (run-time variation of functionality).

Case Naava

Naava is also a very large ICT company (~100,000 employees globally), developing telecommunications solutions and cellular broadband systems.
Consequently, Naava is operating in the same markets as Korte and is one of its main competitors. The STH+ model was applied in an analysis of Naava’s CSE capabilities (Table 12).

Table 12. STH+ analysis in case Naava (Paper II).

<table>
<thead>
<tr>
<th>Company</th>
<th>Dimension</th>
<th>Traditional</th>
<th>Agile</th>
<th>CI</th>
<th>CD</th>
<th>IES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naava</td>
<td>Business</td>
<td>NA</td>
<td>IN</td>
<td>PR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td>NA</td>
<td>IN</td>
<td>IN</td>
<td>PR</td>
<td>PR</td>
</tr>
<tr>
<td></td>
<td>Process</td>
<td>NA</td>
<td>IN</td>
<td>PR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Organisation</td>
<td>NA</td>
<td>PR</td>
<td>PR</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

The five interviewed employees of Naava are developing a cellular network traffic analyser product. A traffic analyser helps system operators to monitor telecommunications system functionality and identify whether a network has any problems and deficiencies related to system performance. Consequently, this analyser can be considered as a complementary service that Naava is offering to telecom operators.

The interviews indicated that practices related to agile were mostly established in the company. However, especially owing to variations between practices and processes in product programs, the “cross-functional feature teams” practice was not applied in all parts of the company. Subsequently, practices related to the continuous integration and continuous deployment steps were considered to exist only in parts of Naava’s organisation. The IES practice, “architecture supporting run-time variations of functionality”, was considered already to exist in some product programs; however, other practices related to business, process, and organisation did not exist in Naava. Interestingly, Korte had also established technical capabilities for run-time variation practice in some development teams but not in other dimensions related to the IES step.

Case Louhi

Louhi is a large company (~13,000 employees) developing solutions for factory process automation and distributed control systems for pulp, paper, and power mills. Embedded systems related to factory automation involve complex distributed software systems connected together with both a local area network and wireless radio technologies. Many parts of the live factory automation production system located in customer premises are not accessible by Louhi because they are
configured behind customer network firewall solutions, and some of them are not connected to the public Internet at all. The STH+ model was applied in the analysis of Louhi’s CSE capabilities (Table 13).

Table 13. STH+ analysis in case Louhi (Paper II).

<table>
<thead>
<tr>
<th>Company</th>
<th>Dimension</th>
<th>Traditional</th>
<th>Agile</th>
<th>CI</th>
<th>CD</th>
<th>IES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Louhi</td>
<td>Business</td>
<td>NA</td>
<td>PR</td>
<td>PR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td>IN</td>
<td>PR</td>
<td>PR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Process</td>
<td>IN</td>
<td>PR</td>
<td>PR</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Organisation</td>
<td>NA</td>
<td>PR</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

The five interviewed employees are developing an embedded SW & HW platform solution for factory automation. Interviews conducted at Louhi indicated that the company was widely applying in-company “early on requirement freeze” and “milestone-driven development” practices. Only one product program was systematically applying practices related to agile and continuous integration steps. Related to CI step practices, system level verification and validation (V&V) was not integrated in the agile development teams; hence, it was a separate organisation specialising in conducting customer- and factory-related systems performance and safety assurance and certifications. Consequently, CD and IES practices, as defined in the STH+ model, were not considered to be feasible for wider adoption in the company. However, according to the interviewees, some parts of a factory automation system, related to user experience, could potentially be improved by also applying continuous deployment and IES practices.

Case Hoilua

Hoihua is a large company (~13,000 employees) that provides a wide range of IT services for several industries, especially for the public governance, telecommunications, consumer electronics, and semiconductor industries. Consequently, the company has experience in working with different technology platforms and customers having different needs for IT systems. The STH+ model was applied in the analysis of Hoihua’s CSE capability (Table 14).
The four employees interviewed were developing the company’s public website. The interviews indicated that a transition towards CSE was mainly happening in distributed teams and project level practices related to agile and CI steps. The interviews did not indicate any product program- or company-level drivers, i.e. activities for moving towards CSE, as a company. Consequently, practices related to the traditional step in the STH+ model were still applied widely in company projects, e.g. “early on requirements freeze” and “customer validation at the end of the project”. However, the interviews indicated that some projects and teams had adopted the “cross-functional team” practice, but overall awareness and competences related to agile practices and CI were not fully established in the company.

Case Latva

Latva is clearly the smallest among the case companies (~600 employees) investigated in this dissertation. However, Latva is developing a wide range of products and platforms and is offering R&D services to other companies, which typically involve radio technologies and wireless data transfer. Latva’s capabilities related to CSE were analysed in two separate studies described in Paper II and Paper III. First, in Paper II, the STH+ model was applied in an analysis of Latva’s CSE capabilities (Table 15). In Paper III, the CRUSOE framework was used to analyse the prerequisites for using CSE in one of Latva’s product programs. The seven interviewed employees were involved in a smartphone platform development project.

---

**Table 14. STH+ analysis in case Hoilua (Paper II).**

<table>
<thead>
<tr>
<th>Company</th>
<th>Dimension</th>
<th>Traditional</th>
<th>Agile</th>
<th>CI</th>
<th>CD</th>
<th>IES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hoilua</td>
<td>Business</td>
<td>IN</td>
<td>TE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td>IN</td>
<td>TE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Process</td>
<td>IN</td>
<td>TE</td>
<td>TE</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Organisation</td>
<td>IN</td>
<td>TE</td>
<td>TE</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Table 15. STH+ analysis in case Latva (Paper II).

<table>
<thead>
<tr>
<th>Company</th>
<th>Dimension</th>
<th>Traditional</th>
<th>Agile</th>
<th>CI</th>
<th>CD</th>
<th>IES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latva</td>
<td>Business</td>
<td>NA</td>
<td>IN</td>
<td>IN</td>
<td>NA</td>
<td>PR</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td>NA</td>
<td>IN</td>
<td>IN</td>
<td>PR</td>
<td>PR</td>
</tr>
<tr>
<td></td>
<td>Process</td>
<td>NA</td>
<td>IN</td>
<td>IN</td>
<td>TE</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Organisation</td>
<td>NA</td>
<td>IN</td>
<td>IN</td>
<td>PR</td>
<td>NA</td>
</tr>
</tbody>
</table>

The interviews indicated that Latva had established capabilities associated with practices related to both agile and CI steps. In addition, practices for the CD and IES steps were also applied in some product programs and teams. Hence, while the company clearly had developed the capabilities for CSE, these capabilities were not systematically applied in all company projects. The interviewees emphasised that CSE applicability is highly dependent on both the customer and the product under development.

The investigated smartphone platform project was considered highly capable of using the CSE approach in product development. However, limitations were related to supply chain and customer processes, which hindered comprehensive application of the continuous deployment and experimentation-driven approaches in product development. The analysis was performed using the CRUSOE framework (Table 8), and it indicated the following key prerequisites for using CSE in the smartphone platform project at Latva:

1. Customer education and motivation
2. Software ecosystem support
3. Supply chain stakeholder support
4. Leadership commitment
5. Process rigour for experimentation
6. Quality assurance process cycle duration
7. Technical debt management
8. Over-the-air updates with minimised breaks in service availability
9. Internal experience sharing and bottom-up strategic planning.

**Summary of key benefits and barriers of using CSE in case companies**

Table 16 summarises the cross-case analysis of the case companies’ key benefits as well as barriers associated with taking steps from traditional product development towards CSE. It should be noted that none of the interviewed companies were considered to have fully achieved continuous deployment and IES capabilities.
Consequently, the interviewees’ opinions on the benefits of fully adopting the IES approach in development should at this point be considered mainly speculative. Case study findings are considered to be congruent with earlier studies on CSE (Claps et al., 2015; Leppänen et al., 2015; Lindgren & Münch, 2015; Olsson et al., 2012; Rissanen & Münch, 2015) that have indicated challenges in the use of continuous deployment, continuous delivery, and continuous experimentation practices.

Table 16. Cross-case analysis on benefits and barriers related to steps towards CSE (Paper II).

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Traditional -&gt; Agile R&amp;D</th>
<th>Agile R&amp;D -&gt; CI</th>
<th>CI -&gt; CD</th>
<th>CD -&gt; IES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Benefits</strong></td>
<td>Short sprints provide the possibility of quickly changing the course of product development.</td>
<td>Provides the ability to build and test products incrementally. Provides high-quality software functionality with production quality.</td>
<td>Customers get fast and incremental delivery of relevant functionality.</td>
<td>The innovation validation is fast.</td>
</tr>
<tr>
<td><strong>Barriers</strong></td>
<td>It is difficult (a complex process) to align different cross-functional teams within the R&amp;D organisations.</td>
<td>There is a lack of team discipline, test-driven development (TDD) and module tests for CI test automation.</td>
<td>The shortening of the V&amp;V cycle is complex and expensive.</td>
<td>Customer feedback is integrated into the short development and business planning cycle. It is difficult to conduct experiments in safety-critical systems.</td>
</tr>
</tbody>
</table>

4.3 A systematic literature review (Paper IV)

This section focuses on following research questions:

*RQ3: What are the impacts of using CSE?*

*RQ4: How is CSE investigated in SE studies?*

The SLR presented in Paper IV extends the related systematic mapping study (SMS) on continuous deployment (Rodríguez et al., 2017). SMS results indicated that many benefits and challenges were reported on continuous deployment. SMS did not provide comprehensive analysis of any other impacts or explanations for either
benefits or challenges associated with CSE. Moreover, Rodriguez et al. considered empirical evidence to be limited owing to the nature of industry reports:

_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_ (Rodríguez et al., 2017).

Consequently, it was considered beneficial to further analyse and re-evaluate the state of the research on continuous deployment and also other related modern release engineering practices (Figure 14).

Figure 14 illustrates the key search terms applied in search of primary studies. To resolve the main research question on the impacts of ARE practices, the SLR included in the analysis only empirical studies performed in a real software development context. The SLR applied Ivarsson et al.’s (Ivarsson & Gorschek, 2010) scale for evaluation of scientific rigour, and studies with low scientific rigour were excluded. Seventy-one studies could be identified as relevant primary studies on ARE practices; however, 33 of these did not specify any research method or data collection approach. Hence, they were considered casual experience reports (industry reports). A little over half of the 71 studies, i.e., 38 empirical studies, described their data collection and research methods to a satisfactory level and could be therefore used for a deeper analysis on the impacts of ARE practices.

![Agile Release Engineering Diagram](image)

Fig. 14. Agile release engineering practices investigated in Paper IV.

Related to RQ4, the SLR investigated research and data collection approaches to ARE practice. The SLR analysis indicated the following four clusters among the 71 primary studies:
a) Studies investigating impacts associated with changing the development practice. E.g. “What factors foster/prevent adoption of the practice?” (22 studies)

b) Studies investigating the prevalence of the practice in software-intensive projects. E.g. “What is the significance of the practice?” (7 studies)

c) Studies quantifying impacts associated with software development success. E.g. “How much has the practice impacted product quality?” (17 studies)

d) Studies describing lessons learned in using the practice. E.g. “How have we implemented the practice?” (33 studies)

The SLR analysis indicated the following four clusters related to empirical data collection approaches:

a) Studies analysing data collected via work products, error reports, log files, etc. (12 studies)

b) Studies analysing data collected via surveys or interviews with stakeholders (23 studies)

c) Studies applying both the A and B approaches for data collection (3 studies)

d) Experience reports with no data collection approach specified (33 studies)

The SMS and SLR (Paper IV) primary study distribution by year (including all 71 primary studies), illustrated in Figure 15, indicates that publication on CSE topics has increased significantly very recently, especially between 2012 and 2015. It should be noted that SMS (Rodríguez et al., 2017) and SLR (Paper IV) applied different strategies in search of primary studies. First, the search strings used were different. Second, while the SLR searched for explicit terms such as “continuous deployment”, the SMS strategy was to find articles that mentioned both “continuous” and “deployment”, but these words did not have to occur one after the other or in this order. Consequently, the primary studies analysed differ between the SMS and SLR. However, both studies clearly indicate that publishing of relevant studies on CSE has increased significantly, especially in 2012, and the number of studies has grown each year until mid-2015, which is the period covered in this study. Thirty-three of the 71 primary studies were published in the 2014–2015 period, and 16 studies were published during the first half of 2015.
While the number of studies on ARE practices has increased over the most recent 2–3 years, there are issues that should be increasingly addressed in future studies. Currently, empirical evidence is largely based on data collected from qualitative surveys and interviews. Qualitative studies can provide insights for scoping future studies as well as providing an overview of the significance and prevalence of the practice in the software development domain. However, these findings often tend to be more vulnerable to subjective preferences and cognitive biases. A synthesis of empirical studies indicated several areas for improvement in the quality of research on ARE practices. Hence, the following improvements in future research are suggested in Paper IV:

- Future research on ARE practices could benefit from a more extensive use of quantitative methodologies from case studies, such as the mining of bug repositories before and after a rapid release model was taken into use by a company. Combining statistical analysis with interviews (methodological triangulation) could also provide for a more reliable analysis of the impacts.
– A more detailed description of the context is important to establish a baseline for meaningful analysis and a comparison between investigated software projects.

– Currently, empirical studies are focused mainly on a generic software supplier point of view. Investigation of ARE practices could benefit from embedding explicit stakeholder points of view on ARE practices, such as a software developer, customer, and technology platform supplier.

– Empirical evidence on ARE practices and their use with respect to experiment-driven development practices is still limited. An investigation of techniques and practices for involving customers in development cycles would also aid in understanding how companies can move towards the use of CSE.

– Several primary studies on ARE practices indicated a close relationship with company-level business and software ecosystem processes. Further investigation of ARE practices and ecosystems would help in pinpointing the most desired ecosystem characteristic from an ARE practices point of view.

Challenges in the adoption of ARE practice were clearly one of the most investigated topics among the primary studies. The research of 22 primary studies focused on the identification of challenges in the adoption of ARE practices, i.e. the change of software development practices towards CSE. The continuous deployment strategy was considered challenging in many primary studies. Consequently, in summarising the most common challenges, 14 prerequisites for applying the continuous deployment strategy were compiled, as follows:

1. All project members understand agile development values and principles in software development.
2. All project members comply with continuous integration practice in software development.
3. The software architecture and system modularity allow for coherent new versions to be produced at any time for CI and testing.
4. Changes in the staging and production environments are tracked, and the process is transparent to all stakeholders.
5. The release schedule, activities, and handovers are synchronised between the internal stakeholders, i.e. developers, testers, and product managers.
6. The release schedule, activities, and handovers are synchronised between the external stakeholders, i.e. contractors, suppliers, and customers.
7. The production system can be updated without interrupting the user.
8. Automated tests can ensure a significant proportion of the system’s core functionality.

9. Third-party applications will work after pushing the changes towards the production environment.

10. Pushing the changes towards the production environment will not break the functionality with external components.

11. The effects on plugins and customer-specific configurations are known before pushing the changes towards the production environment.

12. Customer acceptance testing is integrated into the deployment pipeline.

13. The software branching model is clear to stakeholders, with only short-lived development branches outside the mainline/trunk.

14. Changes do not break the user experience, i.e. the user can fluently adopt the changes and continue using the product/service normally; otherwise, proper notifications and training should be given to the user.

To summarise the SLR findings on RQ3, the primary studies almost unanimously indicated that ARE practices impacted “shorter lead times” and “improved communication within and between development teams”. However, due to the lack and quality of empirical evidence, it is still too early to generalise the ARE practice impacts in the SW development domain. Individual studies in different contexts, which are often described only vaguely, do not allow for reliable syntheses and comparisons between SW development projects, products, and companies.

However, some contextual findings, especially regarding the Firefox browser development, provide interesting indications of the impacts of rapid release. Firstly, it should be noted that it was crucial for Firefox (Mozilla) development to stay competitive with the Chrome browser development and release cycles. Consequently, a market situation such as a competitor transitioning to the rapid release model could illustrate a typical situation and rationale for why software professionals may have to increasingly develop CSE capabilities. Rapid release has required Mozilla to take several actions in SW development practices internally. Meanwhile, a statistical analysis of Firefox bug reports (user-reported bugs) has not indicated a significant change in Firefox software quality, even though the release cycle has changed from 12–16 months to 6 weeks. However, perhaps the most alarming findings are related to open-source community reactions to rapid release cycles. Mäntylä et al.’s (Mäntylä et al., 2014) statistical analysis indicated that the number of volunteer testers in the Firefox open-source community has decreased. This may indicate that the open-source community rejects the usage of rapid release
cycles. Meanwhile, Mozilla has compensated for this by increasing the number of contractors to sustain a satisfactory level of software testing.

4.4 Findings summary

This section summarises the findings for each research question.

4.4.1 RQ1: What are the key aspects of CSE that a reference model must encompass?

The study has indicated the following key aspects:

- Leadership and transformation processes – A model must allow analysis of leadership and integrative management processes. Leadership plays a central role for financing and planning organisational transformation. Leadership has also an important role for fostering a cultural change in organisation.
- Software lifecycle processes and enabling infrastructure processes – A model must provide actionable guidance for CSE practices. However, contextual variations are likely to be required in different domains, such as in service- and product-oriented software development projects.
- Stage-phased transformation in transformation towards CSE – A model must address intermediate steps in transformation towards CSE. Transformation towards CSE is a long-term strategy, and hence, intermediate steps in transformation allow continuous follow-up and measurements related to transformation.
- Interdependencies between strategic planning, development, and operations activities - A model must address value and flow aspects in analysis of how much and how efficiently value is delivered to customer.
- Holistic approach for analysing organisational prerequisites for CSE – A model must allow a holistic analysis of enterprise activities. Holistic analysis allows timely identification of the biggest impediments (bottle necks) related to CSE capability.
4.4.2 RQ2: How does CSE manifest itself in contemporary software-intensive product development?

The study has indicated the following:
- CSE capability is considered as a beneficial strategic direction among software development practitioners. Product quality improvement and operational efficiency improvement were considered to motivate practitioners to adopt CSE capability increasingly in software-intensive product development context. However, many challenges were still identified in the actual use of CSE.
- CSE use in the investigated case companies was often limited in development team and product program level activities rather than the whole organisation’s institutionalised way of working.
- Related business activities to CSE were often considered to be closer to traditional or mixed conventional and agile approaches. While some of the companies had clearly established practices for continuous integration, none of the investigated companies were regularly applying continuous deployment and experimentation in their business level activities.

4.4.3 RQ3: What are the impacts of using CSE?

The study has indicated the following impacts of using CSE:
- CSE impacts are often related to early phase visibility of the product quality and communication among software release stakeholders.
- CSE impacts to shorter lead-times that are achieved by emphasising more frequent integration and smaller releases.
- Organisational uncertainties and challenges related to moving towards CSE are often related to alignment of cross-functional teams, test automation, shortening of V&V cycles, product quality concerns, and integration of customer feedback to business planning cycles.

4.4.4 RQ4: How is CSE investigated in SE studies?

The study has indicated the following patterns in SE studies:
- Number of relevant studies on CSE has increased significantly over the recent 2–3 years.
- Most studies on CSE are considered as industry experience reports.
- Empirical evidence on CSE is largely based on qualitative surveys and interviews.
5 Discussion and conclusion

This section concludes the dissertation and summarises the main results of the study. Threats to research validity, limitations, and future opportunities related to research are also discussed. Figure 16 summarises the research questions and the key contributions of the dissertation.

RQ1: What are the key aspects of CSE that a reference model must encompass?

RQ2: How does CSE manifest itself in contemporary software-intensive product development?

RQ3: What are the impacts of using CSE?

RQ4: How is CSE investigated in SE literature?

5.1 Summary of the results

Based on the findings of Papers I–IV, outlined in section 4, the results of this dissertation are summarised as follows:

RQ1: What are the key aspects of CSE that a reference model must encompass?

The investigation of the LESAT for SW approach indicated variations in industry and company interpretations of the lean development. While MIT’s LESAT approach focused on leadership and enterprise transformation processes, Ericsson’s
approach was more focused on the roles and the development teams’ behaviour. This indicates different ideas and needs related to lean principles and business transformations in companies. Room for contextual variations in a CSE reference model is also considered to be necessary.

The study was inspired by holistic models such as Continuous*, ESAO, and BAPO in the classification of key practices and prerequisites for using CSE. The case company evaluations performed with STH+ and the CRUSOE framework indicated the viability of these approaches for analysing company and development project level capabilities in the use of CSE. CSE addresses a wide range of continuous activities that may require an even more radical “blurring of classical boundaries” between business, development, and operations. Consequently, addressing DevOps and BizDev concepts in CSE is a very useful way to emphasise the continuous interdependencies between development, business, and operations. Hence, CSE reference models should also clearly indicate stakeholder value and flow aspects throughout different functions in product development. The emphasis of locally effective continuous practices could also easily lead to harmful sub-optimisation. This is where lean thinking and related techniques, such as Kanban boards and value stream mapping, could significantly aid in establishing a continuous flow of value.

**RQ2: How does CSE manifest itself in contemporary software-intensive product development?**

Related studies on CSE (Rahman et al., 2015; Rodríguez et al., 2017) indicated a close relationship with cloud computing, e.g. the development of websites and applications that exploit cloud capabilities for data processing. The dissertation conducted case studies mainly in the product-intensive embedded systems domain. The investigated companies applied CSE practices only within some individual teams and product programs, but CSE was not a prevailing and dominant software development approach in the enterprise level.

Challenges in CSE adoption are often related to product-oriented software development context, e.g. in this dissertation, the case studies were conducted for embedded systems in a telecommunications context. This further strengthens the notion of the usefulness of CSE development practices and their applicability in the cloud computing context. The dissertation confirmed challenges in the adaptation of CSE practices in the development of embedded systems. However, the dissertation findings also support the notion of the increasing relevance of CSE in
many software development contexts. The interviewees were mostly positive in their opinions about the anticipated benefits associated with the CSE capability. Nevertheless, contextual adaptations of CSE practices for the embedded systems domain will most likely be needed in all of the investigated companies. Moreover, quality assurance and the related need for test automation were often considered as a bottleneck for using CSE. These contextual limitations could mean that release cycles in embedded systems and physically distributed computing platforms and the multi-vendor systems context may, at least temporarily, have to settle into monthly or weekly release cycles rather than daily or even quicker customer release cycles.

RQ3: What are the impacts of using CSE?

The empirical studies almost unanimously indicated that ARE practices impacted “shorter lead times” and “improved communication within and between development teams”. However, studies have frequently indicated contextual challenges in particular in the adoption of CSE. Existing evidence related to the success of software development in using CSE is largely based on practitioners’ perceptions of how CI and test automation helps in shortening the integration process, lead times, and communication within and between stakeholders. Identified challenges were often related to several dimensions of CSE. The CRUSOE model addressed these dimensions with seven areas related to an ecosystem-driven approach to analysing product development. The SLR (Paper IV) indicated the impacts of ARE practices on various subdomains of software development. The case studies conducted in product-oriented software development indicated that CSE is seen as a beneficial strategic long-term objective for ICT companies. In the very large companies investigated in this dissertation, CSE practices were often already used in individual teams and sometimes even at the product program level. The conducted interviews indicated the CSE dependency on products and stakeholders. Customers and technology platforms especially may have strong impacts on the release and experimentation cycles.

RQ4: How is CSE investigated in SE literature?

An SLR (Paper IV) investigated the CSE topic from the release engineering point of view (agile release engineering practices). The study indicated the novelty of the continuous development concepts in the SE research domain. The number of relevant studies on CSE has increased significantly over the most recent five years.
When performing a deeper analysis on the quality and context of the primary studies, the SLR indicated that almost half of the 71 published empirical studies were experience reports (i.e. casual industry experience sharing), which lack a rigorous description of the research context and applied development practices. This further limits opportunities for the synthesis and accumulation of knowledge related to the impacts of ARE practices and the evaluation of the applicability of the CSE approach in software development. Empirical studies on ARE practices have mainly applied qualitative research approaches, e.g. surveys and interviews on analysing challenges in the adoption and use of CSE.

5.2 Threats to research validity

This section discusses the main threats to the validity of the results presented in this dissertation. Threats to research validity are discussed using a classification scheme presented in (Runeson & Höst, 2008): construct validity, internal validity, external validity, and reliability.

External validity threats are related to situations when the researcher aims for a generalisation of the findings. The main concerns related to external validity are to what extent the findings can be generalised, and to what extent the findings are of interest to other people outside the investigated case (Runeson & Höst, 2008). In this dissertation, the case study findings were used in conjunction with building insight into CSE as a phenomenon and for understanding the common characteristics of CSE theory building, i.e. model design and evaluation.

A reliability threat is concerned with researchers’ dependency on the data and the analysis of the results. Thus, a reliability threat analysis aims to understand the extent to which results are dependent on the researcher. Ideally, research can be repeated by another researcher and produce the same results. To mitigate reliability threats, the interviews and data analysis were conducted collaboratively with two or three researchers. At least two researchers participated in all interviews. While one researcher was responsible for asking questions, another researcher was responsible for taking notes. Immediately after each interview, the researchers shared their observations and notes with each other. This helped them achieve a common insight on case company practices. It also helped the researchers to identify areas that needed more details on company practice and to improve interview techniques for the following interviews. In Paper IV, the SLR primary study selection and assessment steps were duplicated by two researchers to mitigate subjective bias in the SLR findings.
Construct validity addresses threats related to operational measures and how they represent the research questions and the investigated phenomena that the researcher has in mind. Runeson et al. (Runeson & Höst, 2008) states that a potential threat related to interviews is that constructs discussed in an interview are interpreted differently by the interviewer and interviewee. Risks related to construct validity regarding CSE research are substantial, especially because terminology related to various continuous activities (continuous experimentation, continuous integration, continuous delivery, continuous deployment, rapid release, etc.) is not self-evident and clear either for SE researchers or practitioners.

The interviews at the companies aimed at understanding how company practices have changed and how they would have to be further changed in order to move towards CSE. In addition, the interviewees were asked to evaluate benefits and barriers in the transition towards CSE. However, the transition from traditional development towards CSE could take many years. Consequently, the interviewees’ memories about the early stage transition may not be as accurate as they are for more recent events in the company. This situation could involve validity threats related to perceived and anticipated benefits and barriers.

To mitigate risks related to construct validity, before the interview, each interviewee was given an introductory document on research background, goals, etc. However, there was still no certainty about whether the interviewee had actually read and understood the introduction. Consequently, each interview started with an approximately 5–10 min introduction to the N4S program goals and to the research topic. This was considered to help the interviewees to more accurately evaluate and associate their existing development practices with, for example, “real-time value delivery” and “deep customer insight” goals. In addition, the STH model was also briefly explained, and the steps associated with the model were discussed during the interview.

The construct validity threat related to the aggregation of impacts of ARE practices, presented in Paper IV, involves the risk of comparing very different research contexts and implementations of ARE practices. This is generally considered a larger problem related to the reporting of empirical research. If the context and practices are not properly addressed, synthesis of the impacts of ARE practices will also be limited in the following years, as empirical evidence on CSE practices increases.

A typical risk of internal validity threat is related to a situation when the researcher investigates whether one factor affects an investigated factor. In this situation, a third factor of which the researcher is not aware could also affect the
investigated factor. Investigations on the impacts of CSE may very likely entail the risk of internal validity. To analyse factors related to the CSE in companies in perfect isolation from other related factors is virtually impossible. For future studies, this internal validity threat with respect to CSE indicates a need for careful research planning, preparation for measurement, and data analysis.

5.3 Limitations of the research

This dissertation applied SLR and case study methods in conjunction with the DSR approach to investigate CSE. Consequently, the main objective was to design and evaluate models for the abstraction of key CSE concepts and their relationships. The study largely applied existing approaches and models in SE, and hence, the limitations of LESAT for SW, STH+, and CRUSOE are related to the assumptions used in the baseline approaches.

Two related topics on CSE were deliberately not included in this dissertation. The study focused mostly on the reference model design and evaluation related to software development and the business dimensions of CSE. Consequently, the research does not address the operations aspect of CSE research. Operations aspects are elaborated more in DevOps-related studies (L E Lwakatare et al., 2016). In addition, the research does not specify empirical findings on continuous customer feedback and the involvement cycles, which are elaborated by Sauvola et al. (Sauvola et al., 2015).

One of the main limitations in the dissertation is that the LESAT for SW, STH+, and CRUSOE models has been used in conjunction with only a few case companies. Moreover, applying models in different contexts and together with companies where continuous deployment and experimentation practices are institutionalised would be useful for validating assumptions related to CSE practices. Experiences from different companies would allow a more accurate evaluation of compatibility and transferability between different software development contexts.

The interviews involved 4–7 employees from each company. Significantly more interviews and empirical data would be needed to establish a more reliable analysis of companies’ capabilities, barriers, and benefits in the transition towards CSE. Therefore, the results presented in this dissertation should still be considered tentative, aiming for a piloting of models to evaluate CSE capabilities.
5.4 Future research opportunities

This dissertation has indicated various potential directions and topics for conducting future studies on CSE. Some of the most interesting future research areas are discussed in this section. From an industry point of view, the recommended directions for future research activities are considered to be related to a more detailed understanding of the consequences of CSE practices, such as continuous integration (Ståhl, Mårtensson, & Bosch, 2017).

One of the main areas not covered in this dissertation is related to software usage, i.e. customer perceptions and capabilities with respect to the CSE. As stated by Runeson et al. (Runeson & Höst, 2008), study objects in SE research are private corporations or units of public agencies developing software rather than public agencies or private corporations using software systems. However, the user aspect related to CSE would also be an important research topic. Paper III identified the “motivated and educated customer” as a key prerequisite for CSE. In addition, other studies, such as that by Olsson et al., have pointed out the customer aspect in the CSE:

*When moving towards continuous deployment of software, companies need to identify a proactive customer within the ecosystem who is willing to explore the concept. When having identified a ‘lead customer,’ the development organization can start building a continuous deployment culture and capability, in which fast customer feedback serves as direct input to improved software functionality. The lead customer serves as a role model to other customers in the ecosystem and benefits from the opportunity to get new software functionality as soon as the development organization has something to deliver* (Olsson & Bosch, 2014, p. 24).

Related to the customer and end-user viewpoint, it would be useful to investigate how widely in the industry customers expect their suppliers to use CSE or what experiences customers have in the use of CSE. Related to increased and continuous customer involvement in CSE, future research could be focused on identifying customer capabilities, e.g. in a B2B (e.g. SaaS business solutions) and B2C (e.g. consumer products) context.

The STH model addresses the evolutionary steps of a company’s capability in moving towards CSE. However, it is not clear how widely companies have actually experienced this kind of evolution. It is also not clear how many years this transformation would typically take. This leads to larger questions related to the
estimation of the costs and incomes of CSE transformation. An estimation of investments for CSE infrastructure would be a useful element in conducting an evaluation of CSE adoption in companies. The characterisation of desired and non-desired practices could also further help companies in establishing action plans for adoption. Existing experience reports from organisations, such as GAP (Goodman & Elbaz, 2008), Conject AG (Marschall, 2007), Rally Software (Neely & Stolt, 2013), and the NASA Ames Research Center (Trimble & Webster, 2013), indicate a long-term implementation of change activities towards continuous deployment. While the STH model explains the evolutionary steps in the transition from traditional development to the IES way of working, the study cannot confirm whether transitions follow these exact steps. Case Latva, which, according to the analysis, was the most advanced in CSE capability, applied CSE practices selectively, depending on the product and customer case. Consequently, while the capability for CSE requires certain steps, actual business cases define whether not CSE capability is used.

CSE has been described as a development approach used by companies, such as Amazon and Google, which are considered to have unique capabilities to continuously deploy disruptive new product innovations to markets. To summarise the above-mentioned notions regarding CSE, understanding the principles and holistic viewpoint of CSE is important to be able to lead a transformation towards CSE. However, one must also ensure sufficient investment in underlying modern technology enablers and tools that allow the continuous flow of materials and information.

Established ICT companies are addressing the need to adopt capabilities for continuous deployment and an experiment-driven development approach (Järvinen et al., 2014). Hence, the SE research discipline must address this change in theories and models, which could aid in the assimilation of new capabilities, namely the adoption of methods to identify desired and non-desired development practices and to plan and evaluate the results and efficiency, i.e. the speed and direction of change-related investments and activities.

To conclude this dissertation, the CSE clearly addresses multidisciplinary aspects related to the business, innovation, development, and operation of software-intensive products and services. Thus, resolving issues related to the use and transition towards CSE may provide plenty of research opportunities and increasingly address collaboration activities and information sharing, especially between the software engineering, information systems, and business management disciplines.
References


Original publications


Reprinted with permission from IEEE (I), Springer (II and III), and Elsevier (IV)

Original publications are not included in the electronic version of the dissertation.
679. Suoranta, Terhi (2016) Advanced analytical methods for platinum group elements: applications in the research of catalyst materials, recycling and environmental issues

680. Pesonen, Janne (2016) Physicochemical studies regarding the utilization of wood- and peat-based fly ash


682. Ahmad, Muhammad Ovais (2016) Exploring Kanban in software engineering

683. Mustonen, Kaisa-Riikka (2016) Climate change and boreal rivers: predicting present-day patterns and future changes in hydrological regime and its effects on river communities

684. Karppinen, Pasi (2016) Studying user experience of health behavior change support systems: a qualitative approach to individuals’ perceptions of web-based interventions


686. Taulan, Nebojsha (2016) Choreography modeling in embedded systems domain

687. Yliinne, Henni (2017) Herbivory control over tundra carbon storage under climate change


689. Singh, Sujeet Kumar (2017) Conservation genetics of the Bengal tiger (Panthera tigris tigris) in India


691. Hens, Hilde (2017) Population genetics and population ecology in management of endangered species


694. Schneider, Laura (2017) Mechanocatalytic pretreatment of lignocellulosic barley straw to reducing sugars

Book orders:
Granum: Virtual book store
http://granum.uta.fi/granum/
Teemu Karvonen

CONTINUOUS SOFTWARE ENGINEERING IN THE DEVELOPMENT OF SOFTWARE-INTENSIVE PRODUCTS

TOWARDS A REFERENCE MODEL FOR CONTINUOUS SOFTWARE ENGINEERING

Teemu Karvonen