OULU 2016
A 682

ACTA UNIVERSITATIS OULUENSIS

Muhammad Ovais Ahmad

EXPLORING KANBAN IN SOFTWARE ENGINEERING

UNIVERSITY OF OULU GRADUATE SCHOOL;
UNIVERSITY OF OULU,
FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

ACTA UNIVERSITATIS OULUENSIS
A Scientiae Rerum Naturalium 682

MUHAMMAD OVAIS AHMAD

EXPLORING KANBAN IN 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 the Wetteri auditorium (IT115), Linnanmaa, on 25 November 2016, at 12 noon

UNIVERSITY OF OULU, OULU 2016

Acta Univ. Oul. A 682, 2016

Supervised by
Professor Markku Oivo
Doctor Jouni Markkula

Reviewed by
Professor Ivan Porres
Professor Kenichi Matsumoto

Opponent
Professor Pasi Tyrväinen

ISBN 978-952-62-1407-8 (Paperback)
ISBN 978-952-62-1408-5 (PDF)

ISSN 0355-3191 (Printed)
ISSN 1796-220X (Online)

Cover Design
Raimo Ahonen

JUVENES PRINT
TAMPERE 2016

Ahmad, Muhammad Ovais, Exploring Kanban in software engineering.
University of Oulu Graduate School; University of Oulu, Faculty of Information Technology and Electrical Engineering
Acta Univ. Oul. A 682, 2016
University of Oulu, P.O. Box 8000, FI-90014 University of Oulu, Finland

Abstract

To gain competitive advantage and thrive in the market, companies have introduced Kanban in software development. Kanban has been used in the manufacturing industry for over six decades. In the software engineering domain, Kanban was introduced in 2004 to increase flexibility in coping with dynamic requirements, bring visibility to workflow and related tasks, improve communication, and promote the pull system. However, the existing scientific literature lacks empirical evidence of the use of Kanban in software companies.

This doctoral thesis aims to improve the understanding of the use of Kanban in software engineering. The research was performed in two phases: 1) analysis of scientific literature on Kanban in software engineering and industrial engineering and 2) investigation of Kanban implementation trends in software companies. The data was collected through systematic literature reviews, survey and semi-structured interviews. The results were synthesized to draw conclusions and outline implications for research and practice.

The results indicate growing interest in the use of Kanban in software companies. The findings suggest that Kanban is applicable to software development, software maintenance, and portfolio management in software companies. Kanban brings visibility to task and offering status, limits work in progress at any given time gives people greater control over their work and limit task switching. Although Kanban offers several benefits, as reported in this dissertation, the findings show that software companies find it challenging to implement Kanban incrementally.

Keywords: Agile, exploratory research, Kanban, Lean, mixed methods, portfolio management, software development, software maintenance

Ahmad, Muhammad Ovais, Kanban ohjelmistotuotannossa.
Oulun yliopiston tutkijakoulu; Oulun yliopisto, Tieto- ja sähkötekniikan tiedekunta
Acta Univ. Oul. A 682, 2016
Oulun yliopisto, PL 8000, 90014 Oulun yliopisto

Tiivistelmä

Ohjelmistoteollisuudessa Kanbanin käyttö on yleistynyt vuodesta 2004 alkaen. Sillä pyritään tuomaan joustavuutta muuttuvien vaatimusten hallintaan, tuomaan näkyvyyttä työnkulkuun ja toisiinsa liittyviin tehtäviin, parantamaan kommunikaatiota sekä edistämään imuohjauksen hyödyntämistä. Kanbania on käytetty valmistavassa teollisuudessa jo yli kuuden vuosikymmenen ajan. Olemassa olevassa tieteellisessä kirjallisuudessa on kuitenkin esitetty hyvin vähän empiirisiä tutkimustuloksia Kanbanin käytöstä ohjelmistoyrityksissä.

Väitöskirjan tavoitteena on parantaa ymmärrystä Kanbanin käytöstä ohjelmistotuotannossa. Tutkimus toteutettiin kahdessa vaiheessa: 1) Kirjallisuusanalyysi Kanbanin käytöstä ohjelmistotuotannossa ja tuotantotekniikassa ja 2) Empiirinen tutkimus Kanbanin käyttöönoton trendeistä ohjelmistoyrityksissä. Tutkimusaineisto kerättiin systemaattisten kirjallisuuskatsausten, kyselytutkimuksen ja puolistrukturoitujen teemahaastattelujen kautta. Tutkimustulosten synteesin pohjalta tehtiin johtopäätöksiä Kanbanin käytöstä ohjelmistotuotannossa sekä niiden merkityksestä alan tutkimukselle ja Kanbanin käytölle yrityksissä.

Tutkimuksen tulokset osoittavat kasvavaa kiinnostusta Kanbanin käyttöä kohtaan ohjelmistoyrityksissä. Tulosten perusteella Kanban soveltuu käytettäväksi ohjelmistokehityksessä, ohjelmistojen ylläpidossa sekä tuoteportfolion hallinnassa. Kanban tuo näkyvyyttä ohjelmistokehitykseen, niin meneillään olevien tehtävien kuin portfoliotarjoaman osalta. Se myös auttaa rajoittamaan työtehtävien ruuhkautumista ja antaa kehittäjille paremman tavan hallita työtään rajoittamalla työtehtävien vaihtoa. Vaikka Kanbanin käytöllä on mahdollista saavuttaa väitöskirjatutkimuksessa esitettyjä hyötyjä, tulokset osoittavat, että ohjelmistoyrityksillä on haasteita Kanbanin inkrementaalisessa käyttöönotossa.

Asiasanat: Agile, eksploratiivinen tutkimus, Kanban, Lean, monimentelmäinen tutkimus, ohjelmistokehitys, tuoteportfolion hallinta

This thesis is dedicated to my late sister, Nida Maryam

Acknowledgements

"*Know that victory comes with patience, relief comes with affliction, and with hardship comes ease*." Back in 2010, I started my Master's degree program in Software, Systems and Services Development in the Global Environment (GS3D) at the Department of Information Processing Science (former TOL), University of Oulu, Finland. I completed my master's degree in 2011, with an award for being the first fast graduate of GS3D master program. The most valuable aspect of TOL was the easy-to-approach staff members and extra-ordinary people in building student motivation. I really appreciate and thankful to Kari Pankkonen (late) for sincere advices, friendly discussions about research and cultural. Special thanks to Dr. Jouni Markkula who was my master thesis supervisor and later become co-supervisor for my doctoral dissertation.

Later in autumn 2012, roughly four years ago, my journey to PhD degree started which has been a great experience mostly with ups but also some downs to grow me intellectually and personally. The list of people that I acknowledge here is by no way complete; but I would like to thank all of the people who were there during this journey.

First of all, cordial thanks to Prof. Markku Oivo as main supervisor and Dr. Jouni Markkula as co-supervisor, who assisted, motivated and pointed directions through their sheer guidance throughout my research. Professor Markku Oivo is the most professional, supportive, humble and nice person; who always have solution for every bump in this journey. I am deeply thankful to my co-supervisor Dr. Jouni Markkula for patiently supporting my crises, helping to strengthen my research methods approaches, long constructive discussions and supporting me throughout PhD. Their support, encouragement and trust enabled me to carve out, fulfill my tasks and pursue the challenging paths of research. Without their wise counsel and able guidance, it would have been impossible to complete this dissertation.

I would like to extend thanks to members of my follow-up group: Prof. Samuli Saukkonen, Prof. Harri Haapasalo and Dr. Kari Liukkunen. Your feedback steered my research in the right direction. I truly appreciate the support of Dr. Kari Liukkunen, whom I got a chance to work with on research papers outside the scope of my dissertation and learn so many things from you related to research. I would also like to thank Prof. Ivan Porres (Åbo Akademi, Turku, Finland) and Prof. Kenichi Matsumoto (Nara Institute of Science and Technology, Japan), for taking the time and efforts to review my dissertation. Their useful comments and valuable

suggestions helped me to improve quality of my dissertation. I am also very grateful to Prof. Pasi Tyrväinen for his kind acceptance to act as an opponent at my doctoral defense.

I truly appreciate M-Group / M3S research group offers first class environment and it has indeed helped me grow as a better researcher. I express my gratitude to Prof. Pasi Kuvaja for providing opportunity to work in his research projects and making this thesis possible. Pasi always offer special support and friendly attitude throughout this journey. I wish to extend my hearty gratitude to Prof. Burak Turhan, Prof. Petri Pulli, Prof. Seppo Pahnila, Prof. Raija Halonen and Dr. Veikko Halonen for friendly discussions about research, international master's degree programmes, students' supervision and many more topics.

Thanks Anna, Ali, Adrian, Amine, Davide, Elina, Faheem, Harri, Itir, Iflaah, Lucy, Markus, Mirella, Nebojsa, Pilar, Rahul, Sandun, Teemu, Tanja, Valentina and Woub (Alphabetical order) for all the academic discussions, coffees, joint lunches in university and outside work social activities. I would also like to thank vibrant Pakistani community in Oulu, all of my other colleagues from the M-Group / M3S research group, TOL teaching and administrative staff. All of you people were like my family in Oulu. I would like to thanks Davide Anderson and Janice Linden-Reed from LeanKanban, Incorporated for their kindness and support during my stay in USA.

I would not have been able to conduct this research without financial support. I would like to acknowledge DIMECC research projects Cloud Software and Need for Speed, partially funded by Tekes (Finnish Funding Agency for Innovation) as well as participating companies. Moreover, I would like to acknowledge the financial support that I received from University of Oulu Graduate School, Nokia Foundation and Ella and Georg Ehrnrooth foundation grant. Their financial support helps me immensely.

Finally, I would like to give infinite piles of reverence, regards and gratitude to my parents, siblings and Shagul for motivating and pointing me ways to complete my research work efficiently. Despite the immense geographical distance that separate us, they have offered loving encouragement and support throughout my academic years. I am immensely grateful to my parents who provided me immeasurable moral, spiritual and financial support.

Thank you all for contributing to making all this possible. I feel ready to take the next step forward!

Monday, October 13, 2016 Muhammad Ovais Ahmad

Original publications

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

I Ahmad MO, Markkula J, & Oivo M (2013) Kanban in software development: A systematic literature review. IEEE 39th Euromicro Conference on Software Engineering and Advanced Application: 9–16.

II Ahmad MO, Markkula J, & Oivo M & Adeyemi B (2015) Kanban in Industrial Engineering and Software Engineering: A systematic literature review. 10th International Conference on Software Engineering Advances: 234–241.

III Ahmad MO, Markkula J, Oivo M, & Kuvaja P (2014) Usage of Kanban in Software Companies - An empirical study on motivation, benefits and challenges. 9th International Conference on Software Engineering Advances: 150–155.

IV Ahmad MO, Kuvaja P, Markkula J, & Oivo M (2016) Transition of software maintenance teams from Scrum to Kanban. IEEE 49th Hawaii International Conference on System Sciences: 5427–5436.

V Ahmad MO, Lwakatare LE, Oivo M, Kuvaja P, & Markkula J (2016) Portfolio management and Kanban: An empirical investigation with Agile and Lean software companies. Journal of Software: Evolution and Process (Wiley) (In press)

Contents

1 Introduction

In the current fast-paced software business, companies are constantly under pressure to adapt to frequently changing market conditions. Companies are continuously adapting their structures, strategies, and policies in response to new demands. For software companies, the increasingly important questions are how to develop better and cheaper software and how to deliver it faster to fulfil continuously changing customer requirements and market trends. To achieve success in global markets, software companies must account for changing customer requirements during product and service development, while maintaining quality and resisting the urge to solve productivity issues. Until the mid-1990s, plan-driven software development paradigms such as the waterfall and the spiral model dominated the software development landscape (Williams 2012). These paradigms allowed for a disciplined and structured process that met most internal policies and rules of companies.

In the last decade, Agile and Lean became popular in the software industry. The use of the term 'Agile' in software development can be traced back to 2001, when Agile Alliance[1] formulated '*Manifesto for Agile Software Development'*[2]. The manifesto relied on a set of four values and twelve principles (Agile Manifesto, 2001). The four values driving the Agile Manifesto are 'Individuals and interactions over processes and tools', 'Working software over comprehensive documentation', 'Customer collaboration over contract negotiation', and 'Responding to change over following a plan' (Agile Manifesto 2001, Larman 2003). In the context of software engineering, Agile methods include Extreme Programming, Scrum, Dynamic Systems Development Method, Crystal Methods, Feature Driven Development, and Adaptive Software Development (Abrahamsson *et al*. 2003, Gregory *et al*. 2016, Laanti *et al.* 2013, Wang *et al.* 2011). Among them, Scrum is the most popular and widely adopted method (Dingsøyr *et al.* 2008, Rodríguez *et al.* 2012). The use of Agile methods benefits companies in several ways, for example, the ability to respond to dynamic market changes, increased quality, reduced waste, and better predictability (Abrahamsson *et al.* 2009, Abrahamsson *et al.* 2002, Dybå & Dingsøyr 2008, Nurdiani *et al.* 2016, Rodríguez 2013). Despite such benefits, Agile methods have certain limitations and desires. For example, Anderson (2010) explained that technology teams continue to suffer from

[1] https://www.agilealliance.org/
[2] http://www.agilemanifesto.org/

unreliability, businesses have not achieved the expected agility and continue to remain too unresponsive, and costs remain out of control. Agile methods may breakdown in the presence of one or more scaling factor(s) such as distributed Agile development and organisation-wide Agile adoption (Ambler 2009, Maples 2009, Petersen & Wohlin 2009, Wang *et al.* 2011). According to Poppendieck (2007), Agile methods can be applied successfully to software development by understanding Lean.

The origins of Lean can be traced back to the 1940s in Toyota's car manufacturing, which is rooted in the Toyota Production System (TPS) (Womack *et al.* 1990). In manufacturing, Lean focuses on maximizing value and minimizing waste with the purpose of *doing more with less* (Agarwal *et al.* 2006, Radnor & Boaden 2004, Ziskovsky & Ziskovsky, 2007). Kanban is one of the several Lean tools, and it facilitates smooth operation of TPS (Becker & Szczerbicka 1998, Chai 2008, Gross & McInnis 2003, Ikonen *et al.* 2010, Liker 2004).

In the last decade, the software industry and the research community have considered the application of Lean to software development (Wang *et al*. 2012). Lean in software development received increased attention after Poppendieck and Poppendieck (2003) proposed a set of Lean principles, namely, build in quality, create knowledge, defer commitment, deliver fast, respect people, and optimise the whole. Kanban in software development was inspired by Kanban in manufacturing. Kanban implementation facilitates the application of Lean principles to software development (Shinkle 2009). Kanban helps software development teams to work at a sustainable pace, eliminate waste, deliver value frequently, and foster a culture of continuous improvement (Anderson 2010, Shinkle 2009). In 2004, Anderson introduced Kanban to a software development team at Microsoft (Anderson 2010). He described Kanban as follows:

> *Kanban (capital K) is an evolutionary change method that utilizes a kanban (small k) pull system, visualization, and other tools to catalyse the introduction of Lean ideas into technology development and IT operations. The process is evolutionary and incremental.*

In software engineering, Anderson's (2010) definition of the Kanban method is currently well recognised, and the use of Kanban has gained strong practitioner-driven support in the industry (Al-Baik & Miller 2014, Anderson 2010, Cutter 2011, Hiranabe 2008, Hurtado 2013, Shalloway 2011, Shalloway 2010, Shalloway *et al.* 2009, Shinkle 2009). This support is based on Kanban's adaptability (welcomes changes in requirements), visualisation (eases management by visualising

progress), balancing demand against throughput, and minimising work-in-progress to deliver frequently (Anderson, 2010, Hurtado, 2013, Ikonen, 2011, Kniberg & Skarin 2010, Shalloway, 2011). Kanban consultants, promoters, and practitioners have claimed that the use of Kanban brings visibility to work and improves work efficiency, throughput, communication, and collaboration among teams, resulting in rapid software development and continuous delivery to customers (Anderson 2010, Concas *et al.* 2013, Cutter 2011, Hurtado 2013, Ikonen *et al.* 2011, Kniberg & Skarin 2010, Shalloway 2011, Shalloway 2010, Sjøberg *et al.* 2012, Wang *et al.* 2012). However, these claims have not yet been confirmed by empirical studies.

1.1 Research Motivation

In the relevant literature, studies have not drawn distinction between Anderson's definitions of small 'k' kanban and capital 'K' Kanban (Anderson 2010, Al-Baik & Miller 2014, Nikitina & Kajko-Mattsson 2011, Norrmalm 2011, Shinkle 2009, Terlecka 2012). For instance, studies have reported kanban as a pull system to realize continuous production flow; conversely, some findings were more related to the Kanban method (Al-Baik & Miller 2014, Terlecka 2012, Wang *et al.* 2012). A reason for this discrepancy is that in recent years, the move towards Kanban in the software industry has been driven primarily by pioneering practitioners who are familiar with Lean in software engineering. Kanban is fundamentally based on Lean, an older and more mature concept. However, Lean, too, has no common definition in the literature. Womack and Jones (1996) mentioned five principles, while Liker (2004) suggested 14 principles. Moreover, a few researchers have considered Lean as yet another method under the umbrella of Agile (Dybå & Dingsøyr 2009, Highsmith 2002), while others have distinguished Lean as a separate method (Hibbs *et al.* 2009, Wang *et al.* 2012).

Knowledge about methods and processes from other disciplines (such as manufacturing) has limited applicability to software engineering (Münch *et al.* 2012, Mandic *et al.* 2011, Rodríguez 2013). Limited knowledge and understanding of other domains' tools and methods (such as Kanban) and superficial adoption probably do not yield the expected outcomes, resulting in frustration and disillusionment (Ebert *et al.* 2012, Rodríguez 2013).

Kanban practitioners and consultants have been working to establish a comprehensive Kanban methodology for software engineering, but their individual perceptions about the method remain different. The underlying logic is not well understood, and theoretical rationale for labelling Kanban as a Lean tool in software

engineering has not been established (Al-Baik & Miller 2014). As a result, Kanban in software engineering is open to interpretation. For instance, at Microsoft, Anderson (2010) experimented with and developed his own interpretation of Kanban in software development. The small number of available Kanban studies and the dominance of a few authors make it difficult to draw reliable conclusions about the feasibility and adaptability of Kanban in software development. Furthermore, claims about the benefits of Kanban in software engineering have not been validated by empirical studies in software engineering.

1.2 Research Questions

As mentioned above, most current knowledge on Kanban in software engineering comes from practitioners' books, and authors have interpreted Kanban in software engineering in their own ways (Anderson 2010, Burrows 2014, Kniberg & Skarin 2010, Hurtado 2013, Shalloway 2011, Shalloway 2010). Additionally, there is little empirical evidence of the use of Kanban in software development as well as on how Kanban in software engineering is adapted from industrial engineering. To better understand Kanban, this doctoral dissertation analyses a) Kanban in software engineering, b) industrial engineering from where it has emerged, c) Kanban implementation in software companies, and d) practitioners' perceptions of the benefits of and challenges associated with the use of Kanban. Accordingly, the following research questions are formulated (RQs).

RQ1. What is the understanding of Kanban in software engineering based on literature?

RQ.1.1. How is Kanban interpreted in software engineering?

RQ.1.2. How is Kanban interpreted in industrial engineering?

RQ1 focuses on systematic literature analyses of Kanban in software engineering and industrial engineering, the domain in which it originated. No systematic review of Kanban in software engineering research has previously been published. This means that practitioners and researchers have to rely on practitioner books to get an overview. To fill this gap, RQ1.1 investigates the current understanding, knowledge, and state of practice of Kanban in software engineering literature as well as its benefits and challenges. RQ1.1 identifies the needs and opportunities for future research in this area.

As Kanban originated in the manufacturing industry, RQ1.2 analyses Kanban in the industrial engineering context and compares it with Kanban in the software engineering context. This will help to understand Kanban features from industrial engineering field literature and learn how its basic idea can be adapted to the field of software engineering. RQ1 is motivated by the need to understand the differences between the two, which, to the best of our knowledge, has not been done systematically.

RQ2. How is Kanban used in software companies?

RQ2.1 How is Kanban used in software development?

RQ2.2 How is Kanban used for software maintenance?

RQ2.3 How is Kanban used for portfolio management?

There is limited empirical evidence of Kanban use in software companies. In this regard, RQ2 is formulated to provide first-hand industrial insight into how Kanban is being used in software companies. RQ2.1 has three main goals: (1) to seek up-to-date knowledge of the current state, trends, and motivation factors of Kanban use in software companies; (2) to identify the obtained benefits and challenges faced with Kanban use; and (3) to investigate how the identified challenges in Kanban use can be addressed. Additionally, RQ2.1 compares the claimed benefits and challenges of Kanban in software development with the findings of RQ1. RQ2.2 and RQ2.3 provide an in-depth understanding of the use of Kanban in software maintenance and portfolio management in software companies.

1.3 Overview of Research Design

The research approach presented in this dissertation is exploratory in nature (Wohlin & Aurum 2014). This dissertation used multiple research methods to address the research problem from different perspectives in order to strengthen the overall contribution. The research was carried out in the following two phases:

- **Phase 1:** Literature analysis. A systematic literature review method was adopted for analysing the literature on Kanban in software engineering and industrial engineering. The outcome of Phase 1 answers RQ1.
- **Phase 2:** Empirical studies. In Phase 2, three empirical studies were conducted with Finnish software companies that participated in the Cloud Software

Program[3] and Need for Speed (N4S) Program[4]. The outcome of Phase 2 provided answers to RQ2. First, an online survey was administered to explore the Kanban-related trends and the use of Kanban in software development. The survey verified the benefits of Kanban and the challenges faced by software companies, which were found in Phase 1. The outcome of the online survey provided answers to RQ2.1. Further, two qualitative studies were performed to explore Kanban in practice in software maintenance and portfolio management in software companies. These two studies resulted in two papers that answer RQ2.2 and RQ2.3. Finally, the results obtained in Phases 1 and 2 were synthesised to draw conclusions and examine their implications for research and practice.

1.4 Structure of Dissertation

This doctoral dissertation is based on five original research papers (cited as Papers I, II, III, IV, and V). Each paper contributes towards improving our understanding of Kanban in software engineering.

Papers I and II contain literature analyses of Kanban in software development and industrial engineering. Among other findings, Paper I provides understanding of Kanban in software engineering and the implications of its use. It motivated the rest of the studies in this dissertation. Paper II analyses literature pertaining to Kanban in industrial engineering, the domain from which it originated, and highlights similarities with Kanban in software engineering. Among other findings, Paper II highlights other variations of Kanban that can be used in software engineering.

Papers III, IV, and V focus on studying Kanban empirically in software companies. Paper III reports Kanban practitioners' experiences and confirms the findings of Paper I. Paper IV explains how software maintenance teams benefit from Kanban in their work, and Paper V sheds light on the use of Kanban for portfolio management in software companies. The findings of these studies

[3] The Cloud Software Program (2010–2013, http://www.cloudsoftwareprogram.org/) is a Finnish industry-driven research program that includes 22 industrial and eight research participants. The *Cloud Software Program* has the largest volume in terms of budget and companies' involvement in the history of information technology research in Finland.

[4] Need for Speed (2014–2017, http://www.n4s.fi) aims to create a foundation for Finnish software-intensive businesses in the new digital economy. N4S consists of 13 large industrial organisations, 16 small and medium-sized enterprises, and 11 research institutes and universities.

describe how software companies implement Kanban in practice. Each study reports the benefits of and the challenges associated with the use of Kanban, along with possible ways to overcome the identified challenges.

As shown in figure 1, this dissertation consists of five chapters: introduction, background & related work, research design, original research papers, and discussion and conclusion.

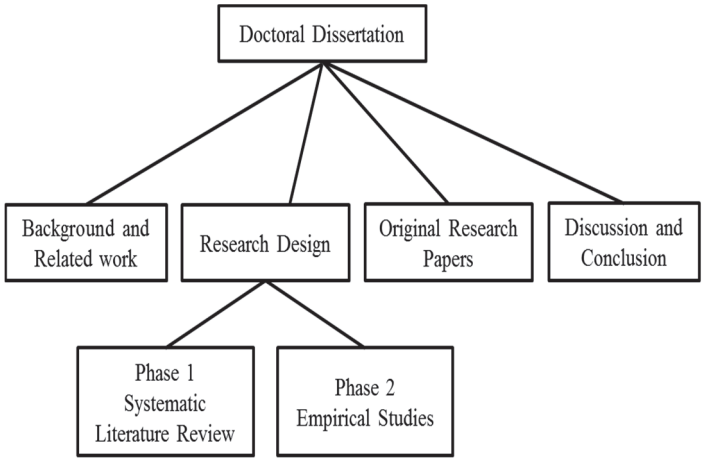

Fig. 1. Dissertation layout.

The remainder of this dissertation is organized as follows. Chapter 2 reviews the backgrounds of Lean and Kanban from their origins, as well as their use in software companies. Chapter 3 presents the research design, including a detailed description of the research methods applied in data collection, data analysis, and reporting. Chapter 4 summarises the contributions of the five original publications included in this dissertation according to research questions that have guided this work. Chapter 5, discussion and conclusion, provides an overview of validity aspects, answers the research questions, and discusses implications for research and practice.

2 Background and Related Work

This chapter discusses Lean and Kanban from the manufacturing and the software engineering perspectives. Sections 2.1 and 2.2 present overviews of Lean and kanban from the manufacturing viewpoint, because the concepts originated from the manufacturing domain. Sections 2.3 and 2.4 explain Lean and Kanban in the context of software engineering. Finally, section 2.5 discusses studies on Lean and Kanban in software engineering and summarises the research gap in the literature.

2.1 Lean in Manufacturing

The Lean concept can be traced back to the 1940s. Lean was first devised as TPS in Toyota's car manufacturing division (Womack *et al.* 1990). Japanese researchers Sugimori, Kusunoki, Cho, and Uchikawa (1977) described TPS as being based on two concepts: 'automation with a human touch' and 'Just-In-Time' (JIT) production (Ohno 1988)[5]. 'Automation with a human touch' means when a problem occurs on a production line, the production line is stopped immediately to prevent the production of defective items. JIT aims to have each process in a production line produce only those items that are needed by the next process in a continuous flow. JIT is driven by a pull system, in which a process withdraws the quantities it requires from the preceding process (Ohno 1988). To implement JIT manufacturing at Toyota, Taiichi Ohno developed kanban[6] to control production between processes and to limit stock at hand to the minimum. Additionally, kanban provides opportunity for active participation, so organisations can uncover the full potential of workers and promote respect for individuals. TPS eliminates waste from the production process, such as overproduction, waiting, transportation, over-processing, excess inventory, movement, and defects.

The term 'Lean' was first used by Krafcik (1988) and, subsequently, by Womack *et al.* (1990) in their book *The Machine That Changed the World*. This book documented the Japanese automobile manufacturing processes that comprise TPS. Most researchers consider TPS to be the most successful application of Lean. Lean is widely used, and there are several interpretations (Liker 2004, Morgan & Liker 2006, Ohno 1988, Womack & Jones 1996, Womack *et al.* 1990), as summarized in Table 1.

[5] The original, Japanese edition Toyota seisan hoshiki by Taiichi Ohno was published in 1978.
[6] i.e., index cards (Poppendieck & Poppendieck 2007: 10).

Table 1. Interpretations of Lean in the production context.

Source	Description
Ohno (1988)	TPS is based on two concepts: 'automation with a human touch' and 'Just-In-Time'. The idea is to eliminate waste such as waiting, transportation, over-processing, inventory, movement, and defects. Waste, or 'muda' in Japanese, is everything that consumes resources but does not yield value.
Womack and Jones (1996)	Specify value, value stream, flow, pull, and perfection are the principles of Lean.
Liker (2004)	To guide TPS, Liker proposed 14 principles across four categories. 1. Long-term philosophy − Base management decisions on a long-term philosophy, even at the expense of short-term financial goals. 2. The right process will produce the right results. − Create a continuous process flow to bring problems to the surface. − Redesign work processes to eliminate waste (muda) through the process of continuous improvement—kaizen. − Use 'pull' system to avoid overproduction. − Level the workload (heijunka). − Build a culture of stopping to fix problems to get quality right the first time. Quality should take precedence (Jidoka). − Standardize tasks and process to facilitate continuous improvement and employee empowerment. − Use visual control, so no problems remain hidden. − Use only reliable, thoroughly tested technologies that serve your people and processes. 3. Add value to the organisation by developing your people − Grow leaders who thoroughly understand the work, live the philosophy, and teach it to others. − Respect your extended network of partners and suppliers by challenging them and helping them improve. − Develop exceptional people and teams who follow your company's philosophy. 4. Continuously solving root problems drives organisational learning − Go and observe first-hand to thoroughly understand a situation (Genchi Genbutsu). − Make decisions slowly by consensus, considering all options thoroughly; implement decisions rapidly (nemawashi). 0 − Become a learning organisation through relentless reflection (hansei) and continuous improvement (kaizen).

Source	Description
Morgan and Liker (2006)	Morgan and Liker describe how Toyota applies Lean and established 13 principles structured into three categories: 1. Process − Establish customer-defined value to separate value-addition from waste. − Front-load product development to thoroughly explore alternative solutions while there is maximum design space. − Create a levelled product development process flow. − Use rigorous standardisation to reduce variation, and create flexibility and predictable outcomes. 2. Skilled people − Develop a chief engineer system to integrate development from start to finish. − Organise to balance functional expertise and achieve cross-functional integration. − Develop towering technical competence in all engineers. − Fully integrate suppliers into the product development system. − Build in learning and continuous improvement. − Build a culture to support excellence and relentless improvement. 3. Tools and technology − Adapt technology to fit your people and processes. − Align your organisation through simple, visual communication. − Use powerful tools for standardization and organisational learning.

Rother & Shook (1999) described Lean as continuous identification and elimination of waste from an organisation's processes, leaving only value-adding activities in the value stream. Waste ('muda' in Japanese) has seven sources: overproduction, waiting, transportation, over-processing, inventories, moving, and defective parts and products (Shingo 1989, Ohno 1988). The focus is on reducing waste in human effort and inventory, reaching the market on time, managing stocks, and producing quality products in the most efficient and economical manner (Bhim *et al.,* 2010, Rahman *et al.* 2013). Other terms commonly associated with Lean are continuous improvement, total quality management, world class manufacturing, theory of constraints, and Six Sigma (Stone 2012). Hallam (2003) suggested:

> *…the proper delineation of the terminology should actually contain three terms, one to describe the end state, one to describe the process that achieves the end state, and one to describe the tools used to execute the process.*

In this way, Lean emphasises actions that deliver value to customers through continuous process improvement while considering the organisation's long-term

perspectives. An important point is that Lean alone is not sufficient; the organisation needs a culture that understands it. According to Womack and Jones (1996), Lean is a way of thinking that must be adopted throughout the enterprise. Furthermore, Morgan and Liker (2006) pointed out that Lean requires the integration of design, manufacturing, finance, human resource management, and purchasing for a product.

2.2 Kanban in Manufacturing

Kanban is one way of executing Lean principles. 'kanban' is a Japanese word meaning 'card' or 'signboard'. It was developed by Ohno in the early 1940s to help Toyota Company fulfil its need of working effectively under specific production and market conditions. In manufacturing, kanban hints what, when, and how much to produce. In Toyota, kanban is applied as one part of TPS to achieve or promote improvement (Hiranabe 2008, Shingo 1989).

According to Sugimori *et al.* (1977), there are three main reasons for using kanban: reduction in information processing cost, rapid and precise acquisition of facts (such as production capacity), optimized operating rate to promote activities for spontaneous improvements and limiting the surplus capacity of preceding shops or stages. Table 2 presents Toyota's description of kanban.

Table 2. Toyota kanban description (Hiranabe, 2008).

TPS Kanban	Description
Physical	It is a physical card.
Limit work in progress (WIP)	Prevents overproduction.
Continuous Flow	Provides information about production needs before a line runs out of stock.
Pull	Downstream processes pull items from upstream processes.
Self-Directing	Contains all information on what to do and makes production autonomous in a non-centralised manner and without micro-management. In this way, people can see the status of work at a glance and detect bottlenecks.
Visual	Stacked or posted to visually show task status and progress.
Signal	Visual status signals the next withdrawal or production action.
Kaizen	Visual process flow informs and stimulates Kaizen.

In manufacturing, kanban signals are physical such as cards and flashing lights. The objective is to deliver material JIT to manufacturing workstations and pass information about what and how much to produce to the preceding stage. The production in a given stage depends on the demand from the subsequent stage; in other words, any given stage must produce only the exact quantity required by the subsequent manufacturing stage (Huang & Kusiak 1996, Bell *et al.* 2016). According to Ohno (1988), kanban should be used with caution, and the rules listed in table 3 need to be considered in kanban implementation.

Table 3. Functions of kanban.

Functions of kanban	Rules for use
Provide pick-up or transport information	Subsequent process picks up the number of items indicated by kanban in the preceding stage.
Provides production information	Preceding process produces items in the quantity and sequence indicate by kanban.
Prevents overproduction	No items are made or transported without a kanban.
Serves as a work order attached to goods	Always attach a kanban to goods.
Prevents production of defective products by identifying the process producing defectives	Defective products are not sent on to subsequent processes. The result is defect-free goods.
Reveals existing problems and maintains inventory control	Reducing the number of index cards increases their sensitivity.

According to Liker (2004), kanban represents the ideal state of JIT manufacturing because it provides customers what they want, at the time they choose, and in the desired quantity (Thun *et al.* 2010). It provides real-time information for limiting work in progress (WIP), and monitoring and controlling production process (Ohno 1988, Zhang *et al.* 2011). Other benefits of kanban in manufacturing and industrial engineering are as follows: it creates visual scheduling, improves flow and responsiveness to changes in demand, facilitates high production, prevents overproduction, improves capacity utilisation, and reduces production time and WIP (Gross & McInnis 2003, Gravel and Price 1988, Kumar & Panneerselvam 2007, Zhang 2011).

In summary, Lean and Kanban have penetrated many industries but were first used in the manufacturing domain with the clear goals of empowering teams, reducing waste, optimising work streams, and, above all, keeping markets and customer needs as the primary decision drivers (Ebert *et al.* 2012, Conboy 2009, Leffingwell & Reinertsen 2011, Rodríguez 2013, Womack & Roos 1990). Such penetration has significantly affected the software industry (Leffingwell &

Reinertsen 2011). In a 2010 survey by Forrester, 35% of the organisations polled stated that their primary development methods are based on Agile and Lean (West & Hammond 2010). VersionOne's annual report (2016) showed that during the last decade, the number of companies that scaled and embraced Lean as part of the larger vision to deliver software in faster, easier, and smarter ways, increased. From 2014 to 2015, the percentage of respondents who practiced Kanban jumped from 31% to 39%, which indicates that software companies are increasingly using Kanban.

Lean has a major impact on the competitiveness of organisations through improvements in process efficiency and reduction in operational waste (Al-Baik & Miller 2014, Liker & Hoseus 2008, Magee 2008). Lean in both manufacturing and software engineering focuses on flow and value creation for customers, along with the systematic analyses of processes to identify and remove waste. Successful stories of Lean and Kanban from the manufacturing industry convinced other industries to use kanban, such as aeronautics (Venables 2005), healthcare (Kim *et al.* 2009), retail clothing (Tokatli 2008), and software development (Anderson 2010, Rodríguez, 2014).

2.3 Lean in Software Engineering

In the early 2000s, Lean received considerable interest from the software engineering domain. Lean in software engineering was adapted from manufacturing. According to Mandic *et al.* (2010) and Münch *et al.* (2012), transforming and adopting concepts from one domain to another domain results in limited applicability. For instance, a manufacturing process yields the same tangible product over and over again. In contrast, software development activities are different in that developers typically create something new in each development cycle. Similarly, in manufacturing, waste refers to tangible products, distinguishing this aspect from software development. In manufacturing waste-related elements can be identified: non-value-adding activities, variations (in process quality, cost, delivery), and unreasonableness (overburden) (Ikonen *et al.* 2010a, Poppendieck & Poppendieck 2003, Poppendieck & Poppendieck 2007, Shingo 1989, Ohno 1988). In software development such elements can be interpreted as: partially, extra processes, extra features, task switching, waiting, motion and defects are considered as waste (Ikonen *et al.* 2010a, Nurdiani et al. 2016, Poppendieck & Poppendieck 2003, Poppendieck & Poppendieck 2007). Therefore, the direct

mapping of Lean and Kanban from manufacturing to software development could be dangerous (Hiranabe 2008).

Similar to manufacturing, in software development, there is no common understanding of Lean, resulting in different interpretations and implementations. As a result, software companies use customised interpretations of Lean (Mehta *et al.* 2008). Many popular books have been written by Lean and Kanban software industry experts (Anderson 2010, Larman & Vodde 2008, Middleton & Sutton 2005, Poppendieck & Poppendieck 2003, Poppendieck & Poppendieck 2007, Reinertsen 2009). Poppendieck and Poppendieck (2003) did prominent work on the transformation of Lean in manufacturing to the software engineering domain. Table 4 summarises various interpretations of Lean in the context of software development.

Table 4. Interpretations of Lean in software development.

Author	Lean interpretation
Poppendieck and Poppendieck (2003, 2007, 2009)	Seven principles that guide Lean in software development − Eliminate waste by first understanding value. − Build in quality by testing as soon as possible using automation and refactoring. − Create knowledge through rapid feedback and continuous improvement. − Defer commitment by maintaining options and make irreversible decisions in the last responsible moment when the most information is available. − Deliver fast in small batches and limit WIP. − Respect the people doing the work. − Optimise the whole by implementing Lean across an entire value stream. Seven sources of waste in software development are: partially done work, extra features, relearning, handoffs, task switching, delays, and defects.
Middleton and Sutton (2005)	Interpretation based on Womack and Jones's (1996) Lean principles: Specify value, value stream, flow, pull, and perfection.
Larman and Vodde (2008)	Lean is based on two pillars 1) respect for people and 2) continuous improvement, and 14 principles: management decisions based on long-term philosophy, flow, pull system, Level the work—decreased variability and overburden to remove unevenness, Build a culture of stopping and fixing problems, master norms (practices) to enable kaizen and employee empowerment, simple visual management, Use only well-tested technology, leader-teachers from within, development of exceptional people, help partners become Lean, Go-See, Make decisions slowly by consensus, Become and sustain a learning organization through relentless reflection and kaizen.
Anderson (2010)	Kanban is one way to execute Lean in software companies. It has five principles: visualise workflow, limit WIP, make policies explicit, measure and manage flow, and use models to recognise improvement opportunities.
Reinertsen (2009)	Set of principles of product development flow, including managing queues, reducing batch size, and applying WIP constraints.
Boehm *et al.* (2014)	The proposed Incremental Commitment Spiral Model has four principles based on Lean: Stakeholder value-based Guidance, Incremental commitment and accountability, Concurrent multidisciplinary engineering, and Evidence and risk-driven decisions.

According to Ebert *et al.* (2012), Lean in software development focuses on creating value for customers, eliminating waste, optimising value streams, empowering people, and improving continuously. Lean begins with the simple premise: 'identify

the 20 percent of the code that provide 80 percent of the value and deliver it just-in-time' (Bell *et al.* 2016). All of those things that do not produce value for customers are considered 'waste' (Poppendieck & Poppendieck 2003).

From all interpretations, it is clear that Lean focuses on end-to-end value flow through development, that is, from very early concepts and ideas to the delivery of software features. In this regard, researchers have proposed various tools and techniques to analyse and improve value flow, such as value stream mapping, inventory management, and pull systems (Anderson 2003, Gross & McKinnis 2003, Mujtaba *et al.* 2010). According to Poppendieck and Cusumano (2012):

> *If Lean is thought of as a set of principles rather than practices, then applying Lean to product development and software engineering makes more sense and can lead to process and quality improvements.*

2.4 Kanban in Software Engineering

Kanban enables all aspects of Lean and provides the tools to optimise an outcome for value through focus on flow management as well as waste reduction (Anderson 2010). According to Shalloway *et al.* (2009), Kanban in software engineering is based on the following three beliefs:

- Software development is about managing and creating knowledge.
- Software development processes can be managed and described in terms of queues and control loops.
- Some representation of information that flows through the system is required.

To put these beliefs in practice, Anderson (2010) explained that Kanban in software development has the following philosophy:

- Start with what you do now.
- Agree to pursue incremental, evolutionary change.
- Respect the current process, roles, responsibilities, and titles.

Kanban suggests starting with the actual state of a company's processes and proceeding incrementally on the basis of explicit policies and data to eliminate bottlenecks in the entire value stream. Such incremental changes enable an organisation to experience recurrent improvement (Kaizen culture) with the passage of time and to solve identified problems individually. Kanban creates a visual task management flow that allows for transparency and process monitoring,

where the focus is on commitment to optimising workflow across the entire value stream (from idea initiation until product consumption by customers).

Kanban aims to improve workflow by removing bottlenecks. Such a business-driven approach creates a broader value stream that warrants participation of all stakeholders and an understanding of WIP. Therefore, Kanban maintains that to be efficient, teams must not be overloaded with work. Overloading teams creates waste, lowers quality, and causes delays in task completion. Kanban highlights inefficiencies and challenges teams to focus on resolving issues in order to maintain a steady workflow (Anderson 2010). To visualise workflow, a Kanban board is used as a tool. A typical Kanban board is shown in figure 2.

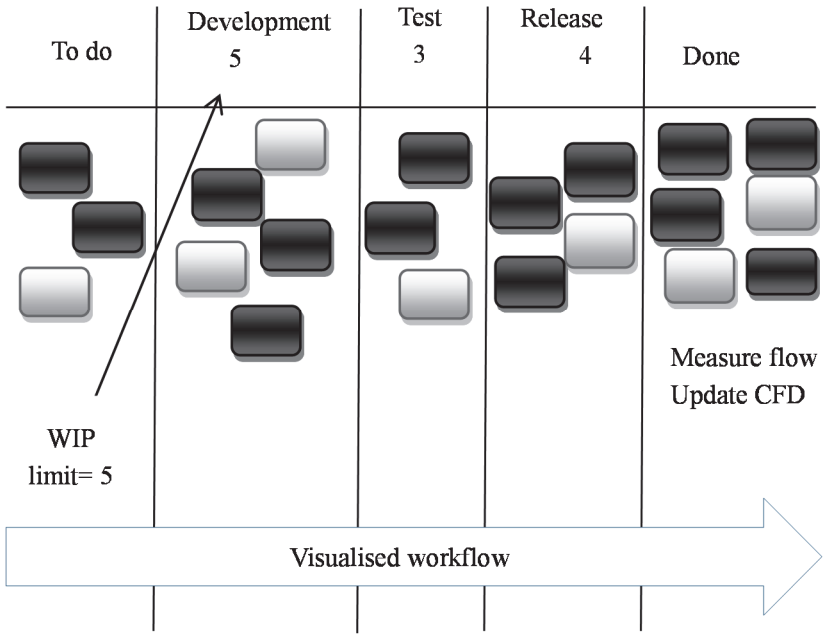

Fig. 2. Kanban board.

From the viewpoint of Kanban implementation, Anderson (2010) identified five key *properties,* which Boeg (2012) called *principles*: visualise workflow, limit WIP, measure and manage flow, make policies explicit, and implement feedback loops and identify improvement opportunities.

2.4.1 Visualise Workflow

Kanban is implemented with a visual board and sticky notes, with each card representing a task or story. According to Anderson (2010):

> *Kanban enables visualisation of wasteful activities and can be used to enable a full Lean initiative within a software, system, product development, or information technology organisation.*

The idea is to visualise workflow and focus on minimal marketable features. A Kanban board is divided into various columns that visualise activity flows in different stages. Cards are used for each task on the Kanban board to show possible statuses. Thus, teams can see the bigger picture of what has been done so far and what needs to be done (Ikonen *et al.* 2011). Additionally, Kanban enforces a steady work sequence according to the pull system. In the case of software development, teams can pull tasks at any time, triggered by available team capacity (Williams 2012). In this way, the development team has greater control of the workload, which can help them deliver in a more continuous manner.

2.4.2 Limit Work in Progress

As a general rule, the greater the amount of WIP, the less is the amount of work that can flow and the slower is the pace of work completion, further increasing the risk of rework and cost increase (Bell *et al.* 2016). Workflow can be optimised by limiting the WIP in each activity column to the maximum number of tasks that can be pulled into the column. This enables software development teams to stay focused and reduce the amount of multitasking.

According to Bell *et al.* (2016), smooth flow produces the transparency and visibility need to coordinate efficient flow of work. By visualising workflow and enforcing WIP limits, bottlenecks in the process become visible, which ensures optimal resource usage. Anderson (2010) pointed out that unilateral declaration of WIP should be avoided; it should be agreed upon by consensus among up- and down- stream stakeholders and management.

2.4.3 Measure and Manage Flow

Kanban focuses on maintaining continuous and smooth flow of work to quickly deliver value to customers. The shorter the lead time, the greater is the value

delivered to customers. In this regard, the Lean concept offers a myriad of tools to analyse and improve value flow, such as value stream mapping, inventory management, and cumulative flow diagrams (Anderson 2003, Anderson 2010, Petersen & Wohlin 2011, Mujtaba *et al.* 2010, Gross & McInnis 2003). A cumulative flow diagram shows the collective number of tasks in each phase of software development. Value stream mapping visualizes the development life-cycle by showing processing times and waiting times. Measurement and models are used to facilitate process-change decisions and optimise processes to decrease lead time.

2.4.4 Make Process Policies Explicit

Explicit process polices and workflow visualisation help organisations to understand and make the right adjustments to processes. Kanban mandates making work policies explicit so that both development teams and management can apply them consistently and see cause and effect whenever they make a change (Cutter 2011). For example, consider the policy that whenever a team member is free, the member should pull a high-priority ticket from the Kanban board. As a result, teams self-organise and manage task flow by themselves. Another example of policy is production stability takes priority over quality assurance bug fixing; both take priority over new development. Anderson (2010) explained that such policies should be discussed openly and revised according to circumstances.

2.4.5 Implement Feedback Loops and Identify Improvement Opportunities

Kanban promotes regular feedback, direct communication, and experiment-based evolution. A lower WIP limit helps teams shorten feedback loops within the process, which causes them to check what has been built and helps streamline workflows. Continuous improvement opportunities can be identified via value stream mapping (Zang 2011).

2.5 Studies on Lean and Kanban in Software Engineering

Lean and Kanban are used in various types of works in software companies. Section 2.5.1 summarises the studies related to Lean and Kanban transformation in software companies and its effects. Section 2.5.2 reports on studies that attempted to analyse the applicability and effectiveness of Kanban by using simulation.

Sections 2.5.3 and 2.5.4 review Kanban implementation studies in software development and software maintenance teams. Finally, section 2.5.5 discusses studies on Kanban implementation in management-related activities such as portfolio management.

2.5.1 Lean and Kanban Transformation

Peter Middleton conducted three studies on Lean transformation and implementation in software engineering. The first study involved two industrial case studies, and the action research method was used (Middleton 2001). Two different teams were formed in the company, one comprising experienced developers (case X) and one comprising developers with relatively less experience (case Y). The results revealed that errors became visible and developers fixed them right away. However, in the long run, the number of errors dropped dramatically. Nevertheless, the teams were unable to sustain their use of Lean owing to organisational hierarchy, traditional promotion patterns, and the fear of forcing errors into the open.

In the second study, Middleton & Sutton (2005) reported on the journey of an American software company (Timberline, Inc.) that had been practicing Lean in their daily work for two years. Before the Lean initiative, the company was facing a number of challenges in traditional software development, such as project resource allocation and tracking, lack of predictability in terms of project completion timeframes, high development costs, and excessive review meetings. The Lean initiative helped the company tackle these challenges. The majority of the company's people supported the Lean ideas and thought they could be applied to software engineering; however, 10% of them were not convinced of the benefits of Lean in software development.

In the third study, Middleton and Joyce examined how the Lean concept could be applied to a software project management team at BBC Worldwide (Middleton & Joyce 2012). With Kanban boards and strict WIP limits, which ensured transparency in detecting bottlenecks in the flow, the team was able to estimate individual and team work efficiently. Over a one-year period, consistency of delivery increased by 47%, and defects reported by customers decreased by 24% (Middleton & Joyce 2012).

Ericsson R&D Finland undertook a comprehensive transition to Lean in 2010. Rodríguez *et al.* (2013) conducted a case study at Ericsson R&D Finland to explore the implementation of how Lean principles in software development. Their results

revealed that the company improved considerably in term of product quality, customer satisfaction, and transparency within the organisation. However, the authors also found that attaining smooth flow, transparency, and creating a learning culture were challenging tasks.

Nikitina & Kajko-Mattsson (2011) reported on a software process improvement effort in a company, realized by transitioning from Scrum to Kanban. After the transition to Kanban, the company solved various problems, for instance, low motivation among developers, confusion with respect to the definition of 'done', insufficient communication of requirements, and low prioritization of technical backlog items. However, a few new problems were introduced by the transition, for example, less control over releases, and extensive feature list requiring long development time. Additionally, Sjøberg *et al*. (2012) conducted a case study to demonstrate to compare the effects of Scrum with those of Kanban by analysing the data of more than 12,000 work items collected over the years 2009–2011 from a medium-sized software company. The results showed that switching from Scrum to Kanban halved the company's lead time, reduced the number of weighted bugs by 10%, improved productivity for project backlog items by 21%, and reduced the number of bugs by 11%.

2.5.2 Simulation Studies about Kanban in Software Development and Software Maintenance

Cocco *et al.* (2011) conducted a simulation study to analyse the dynamic behaviour of Kanban and Scrum adoption in comparison to that of a traditional software development process such as the waterfall approach. The simulation results illustrated that Kanban helped control and manage workflow effectively while minimising lead time. Similarly, Anderson *et al.* (2011) assessed the effectiveness of WIP limits and visualized the flow and organisation of work. Their simulation results validated the WIP limit and yielded insights on what really happens under different development settings. For instance, a WIP limit results in a constant flow of features, whereas the lack of a WIP limit results in a more irregular flow of features.

Concas *et al.* (2013) developed a process simulator to simulate software maintenance processes that 1) use WIP limits and 2) do not use WIP limits. In both scenarios, the authors collected real maintenance data from a Microsoft project and from a Chinese software firm. The results revealed that Kanban helped reduce the

average time needed to complete maintenance requests and that WIP limits can increase the efficiency of software maintenance.

Turner *et al.* (2013) described an example of the implementation of Kanban in a large healthcare system. In large operational systems, where schedules are rarely stable, it is challenging to comprehend the status of capability development. A number of factors are involved, such as the size and complexity of capabilities, unexpected changes in priorities, depth of supplier chains, variety and availability of special engineering resources, contract structure, and the inherently complex nature of such operations. The authors elaborated that under certain circumstances, concepts such as WIP and capacity to maximise flow through a process can be applied to systems engineering and development processes. The findings of a simulation highlighted that Kanban and the pull-system enhance visibility and flow in such complex systems.

2.5.3 Lean and Kanban in Software Development

Ikonen *et al.* (2010a) studied waste in a Kanban-driven software development project in an experimental setting called the Software Factory. 'The Software Factory is an experimental laboratory that provides an environment for research and education in software engineering' (Fagerholm *et al.* 2013). Software Factory provides students with a realistic environment where they can improve their learning experience by gaining insights into the conduct of real-life software projects, which are characterised by close customer involvement, intensive teamwork, and the use of modern software development tools and processes (Ahmad *et al.* 2014a, Fagerholm *et al.* 2013, Ikonen *et al.* 2010a).

The results showed that Kanban implementation helped in the identification of waste, as defined by Poppendieck and Poppendieck (2003). However, the results also showed that finding waste did not contribute significantly to explaining project success. Eliminating waste is only part of a project's operational efficiency which is supported by Kanban. As a visual aid, Kanban helps control software project activities, determine the most important tasks, reveal waste (such as partially done work, non-value-adding work, and task switching), save resources, and adapt to continuously changing situations. Further, Ikonen *et al.* (2011) investigated how Kanban influences software development project work. Their results showed that visualisation helps teams control project activities in a coherently flexible way by relying on team members' intuition. Visualisation was even found to motivate team members. However, the study claimed that Kanban is insufficient for managing all

dimensions of software projects. For instance, Kanban helps identify bottlenecks and detect potential problems as they emerge. Moreover, it is difficult to visualise large and complex system projects in their entirety with a simple Kanban board alone.

Andrezak and Schiffer (2010) reported on the successful use of Kanban in a Europe's largest online automotive market. The company first implemented Kanban in product maintenance, and then throughout the company, rapidly. Kanban enabled the company to release products daily, ensure smooth running of development tasks, and create an effective value chain on the technical level. Furthermore, the company is managing their product portfolio with Kanban, and all product development activities are steered through the enterprise Kanban board, which balances needs and features.

Large-scale distributed software development projects face many challenges such as hierarchical requirements, large team size, and workflow management (Tripathi et al. 2015). A few studies have shown that Kanban helps address such challenges by using WIP limits to manage flow and establish visibility of requirements by using a visual signalling system (Anderson 2010, Kniberg 2011, Tripathi et al. 2015). In this regard, Tripathi et al. (2015) conducted a case study to explore the challenges associated with the scaling of Kanban in large, multisite organisations. The results show that in a multisite environment, an electronic Kanban board helps smoothly plan and execute software development activities. It is recommended to set WIP limits for product backlog and development teams by mutual agreement.

An experience report from Rally Software described their transition journey to continuous delivery with Kanban (Neely & Stolt 2013). The results show that Rally Software achieved greater control and flexibility over feature releases, fewer defects, easier on-boarding of new developers, less off-hours work, and a considerable uptick in confidence.

Ahmad et al. (2016 b) conducted a study with Kanban practitioners to investigate the factors that users perceive to be important for Kanban use. Their study contained 146 responses from 27 different organisations, with all respondents being experienced in using Kanban. The results show that practitioners consider organisational support and social influence to be important determinants for Kanban use. Additionally, the perceived reported benefits of using Kanban are bringing visibility to work, helping to reduce WIP, and improving development flow.

2.5.4 Lean and Kanban for Software Maintenance

Research on software maintenance has focused on traditional software engineering techniques such as modelling, estimation, risk management, statistical process control, quality, metrics, post mortem, and testing (Heeager & Rose 2015). Researchers have paid scant attention to the optimisation of standard software maintenance work processes by using Lean and Kanban. In one study, Ericsson implemented Kanban in telecom product maintenance (Seikola *et al.* 2011). The authors used Kanban to implement Lean principles. They strictly followed the pull system, empowered teams, and introduced WIP limits. The results showed that Kanban increased the visibility of work, drastically increased the level of teamwork, encouraged engineers to be more willing to perform tasks beyond their comfort zone, and contributed to the emergence of a continuous improvement mindset.

ASR Insurance is a large insurance company in the Netherlands. The company transitioned IT maintenance and operations from a more traditional approach to Kanban (Maassen & Sonnevelt 2010). It adopted the underlying principles — make work visible, limit WIP, and help the work to flow. The results showed that Kanban helped in enhancing understanding and cooperation between developers working on different technologies, as well as between the developers and the testers. Similarly, Fundamo, a mobile financial services and products provider, adopted Kanban to control their customer support issues (Greaves, 2011). To measure cycle time and make predictions, the teams used cumulative flow diagrams from day one and recorded the number of issues in each column at the same time on each day after their daily meeting. Kanban helped the teams to quantify both their demand and throughput, and ensured these remained balanced by changing explicit policies.

2.5.5 Lean and Kanban for Management

Ikonen (2010b) conducted an experiment with two directive leadership settings to find the differences between the waste produced, its causes, and effects. The results showed that Kanban allowed for working without a formal project manager from the viewpoint of avoiding waste, but also that insufficient directive leadership created waste (Ikonen 2010, Ikonen 2011). Furthermore, Ikonen (2010b) claimed that the amount and significance of waste can be reduced with the right leadership in self-organized teams of Kanban software development projects.

Software companies use various portfolio management approaches such as Lean portfolio management, Agile portfolio management, program portfolio

management, and the scale Agile framework (Kalliney 2009, Kettunen & Laanti 2008, Laanti 2008, Leffingwell 2010, Rautiainen *et al.* 2011). However, software companies continue to face various problems in portfolio management, such as the lack of visibility of offerings about to enter the development pipeline and their prioritisation (Rautiainen *et al.* 2011).

Leading Lean and Kanban practitioners (e.g., Anderson & Roock 2011, Leffingwell 2010, Shalloway 2011) have claimed that Kanban is the easiest tool to use for portfolio management. According to Shalloway (2011), Kanban provides visibility and helps management make appropriate decisions about tasks based on business value. Laanti and Kangas (2015) reported on the benefits of Kanban in YLE, Finland's national broadcaster. Specifically, the authors noted that Kanban brings visibility and awareness in all ongoing development activities, facilitating easy management of project dependencies and the pipeline. Additionally, Wijewardena (2011) experimented with Kanban in the human resource department of a mid-sized, offshore software development company (Exilesoft). Before Kanban implementation, the human resource department was unable to respond on time to the demands of various projects, which created negative stress and frustration throughout the company. The results showed that Kanban supported the everyday planning required for human resources department functions, brought visibility to their work, and prevented them from over-committing. However, the existing scientific literature lacks empirical evidence of Kanban use for portfolio management.

All the above studies have a number of limitations such as generalizability issues owing to the low number of participants and the lack of description of how various techniques were applied in practice, which makes the findings difficult to interpret. These studies have reported various challenges in Lean transformation, such as making relevant changes to organisational roles, metrics, incentive programs, and deep changes in the way an organisation is managed, including long-term commitment from upper management.

Pernstål *et al.* (2013) conducted a systematic mapping study on Lean in software development by examining the literature published between 1999 and 2010. The study concluded that research on Lean software development is in its nascent state, and 'there is very little support available for practitioners who want to apply Lean for improving large-scale software development' (Pernstål *et al.* 2013). Furthermore, Al-Baik and Miller (2014) reported that there is growing interest and wider application of Kanban in software companies. The study reported a number of challenges in the use of Kanban, such as the absence of guidelines.

There is a lack of empirical studies pertaining to software companies, which makes it difficult to validate or refute the claims associated with Kanban in software engineering. These challenges create difficulties for individuals who plan Kanban implementation. Overall, the results showed that Kanban may be beneficial in the software engineering context. Thus, further exploration of Kanban is needed to better understand and observe its effects in software companies.

3 Research Design

The research conducted in this dissertation falls under the umbrella of exploratory empirical research aiming to obtain a better understanding of a given phenomenon. According to Shull *et al.* (2008), empirical research is investigation of a phenomenon through direct or indirect observation or experience (Creswell 2009). According to Wohlin & Aurum (2014), deductive and inductive are two common ways of reasoning in empirical software engineering. The research in this dissertation is inductive. Inductive research induces generalisations from real-life observations, while deductive research attempts to verify a constructed theory or hypothesis through observation in order to gain a higher level of insight (Basili 1993). In inductive software engineering research, the investigator attempts to understand software processes, products, people, and environments (Basili 1993).

An exploratory research approach is applied when little information is available on a given topic and the investigator aims to gather insights about the problem (Collis & Hussey 2009, Wohlin & Aurum 2014). Exploratory studies are essential to obtain a better understanding of a less clear phenomenon and establish guiding principles for further research. In this regard, Kanban is a new phenomenon under the Lean umbrella in the field of software engineering. Exploratory research can be both qualitative and quantitative in nature (Runeson & Höst 2009, Wohlin & Aurum 2014). However, each research method has its own weaknesses. As such, a viable approach is to use multiple methods to compensate for the inherent weaknesses of the individual methods (Brewer & Hunter 2006, Creswell 2009, Mandić *et al.* 2009, Wohlin & Aurum 2014, Wood *et al.* 1999). This method triangulation helps achieve more credible results by using different methods. Mixed methods is an emergent research methodology that involves collecting, analysing, and integrating quantitative (e.g., experiments, surveys) and qualitative (e.g., focus groups, interviews) research. Mixed methods combine the strengths of both qualitative and quantitative research methods to overcome the weaknesses and limitations of the individual methods (Creswell 2009, Runeson & Höst 2009). Typically, the results of qualitative study are used to support the findings of quantitative study (Easterbrook *et al.* 2008).Easterbrook *et al.* (2008) wrote that:

> *While mixed method research is a powerful approach to inquiry, the researcher is challenged with the need for extensive data collection, the time-intensive nature of analysing multiple sources of data, as well as the requirement to be familiar with both quantitative and qualitative forms of research.*

In this dissertation, data was collected using a systematic literature review, qualitative methods (i.e., semi-structured interviews), and quantitative methods (i.e., online survey). Research design was divided into two phases. In Phase 1, we conducted a systematic literature review, while in Phase 2, we used the mixed methods approach. Figure 3 shows the two research phases.

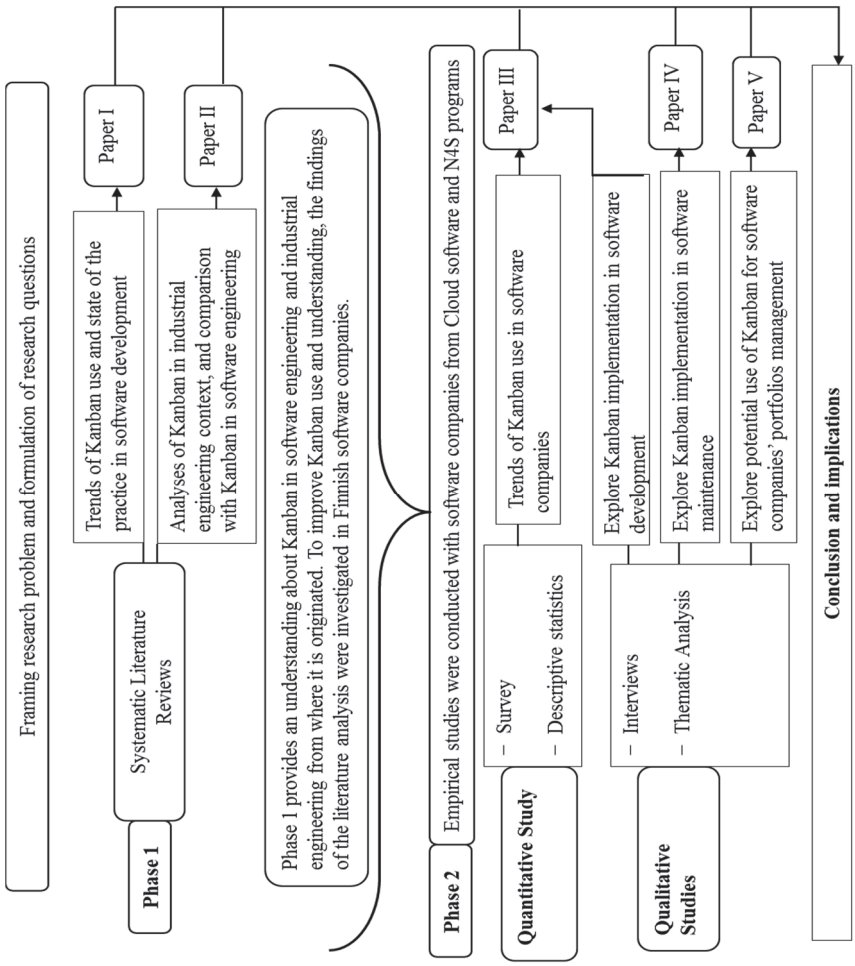

Fig. 3. Two phases of this research.

In the following section, Phases 1 and 2 are described in detail. Each phase is driven by a main research question and a set of sub-questions. Each sub-question is addressed by a separate paper.

3.1 Phase 1: Systematic Literature Review

Systematic Literature Review (SLR), also called systematic review, is considered one of the most important research methodologies in evidence-based software engineering. It is *a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest* (Dybå *et al.* 2005, Kitchenham 2004). According to Babar and Zhang (2009), the majority of reported SLRs in software engineering have been carried out according to the following guidelines put forth by Kitchenham and Charters (2007) for the following primary reasons:

- To summarise existing evidence concerning a treatment, technology, or phenomenon;
- To identify gaps in current research and provide suggestions for further investigation;
- To minimise the level of bias that may be prevalent in ad-hoc literature surveys; and
- To appropriately position new research activities.

3.1.1 Data Collection

In Phase 1, we performed SLRs based on the guidelines of Kitchenham and Charters (2007). The review processes consisted of several phases, including planning, conducting, and reporting. The SLRs were carried out by a team of three researchers. Figure 4 shows a pictorial representation of the SLR process.

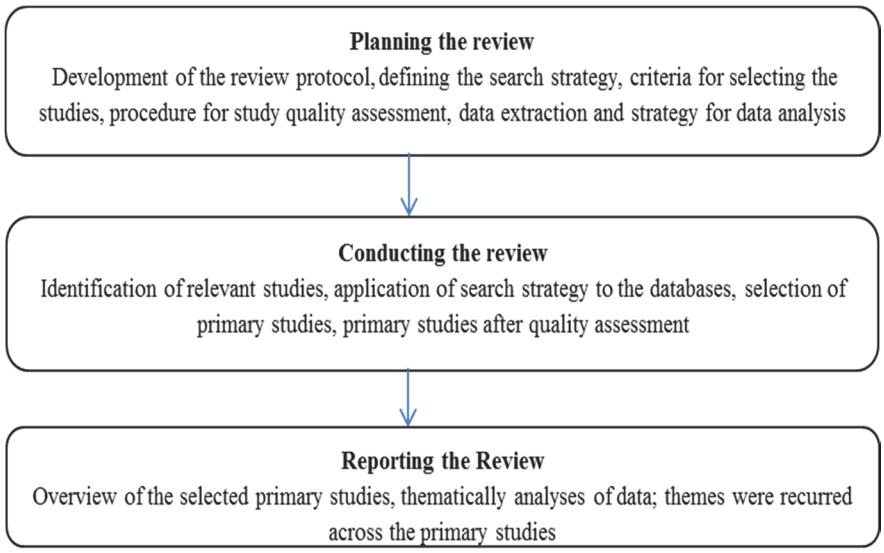

Fig. 4. Steps in SLR.

Table 5 lists the research questions related to Phase 1, and data collection technique used in each study.

Table 5. Research questions, papers, and methods used in Phase 1.

Research question	Paper No.	Title	Data collection technique and participants	Number of primary studies	Studies identified based on search strategy
RQ1.1	I	Kanban in software development: a systematic literature review	Systematic literature review	19	1828
RQ1.2	II	Kanban in industrial engineering and software engineering: a systematic literature review	Systematic literature review	52	1552

The first SLR analysed Kanban in software engineering to provide an overview of the trends in the use of Kanban in software development and future research needs. Additionally, the first SLR summarised practices for smooth implementation of Kanban in software development.

The initial search yielded 1828 papers. The papers were examined against the defined criteria. For example, the search strategy included the term ‘Kanban’, which resulted in the inclusion of several papers about Kanban in manufacturing. These papers were excluded because they were outside the focus of the SLR, leaving 492 papers, which were further assessed with regard to quality and relevance to this literature review. This narrowed down the set to 79 papers, which were evaluated based on the following criteria: objective of study, context description, research design, data collection and analysis, justification of findings, result applicability, passing the minimum quality threshold, and use of references. Of the 79 studies evaluated, 19 were finally accepted and included as primary studies in the first SLR.

The second SLR was conducted to explore Kanban in industrial engineering, the domain in which it originated, and compare it with the Kanban in software engineering. The study discussed differences and similarities between Kanban in software engineering and Kanban in industrial engineering.

The search strategy yielded 1552 papers. These papers were analysed to remove duplicates, leaving behind 1366 papers. These papers were further analysed by two researchers to determine their relevance. Only studies published in English were included. Editorials, prefaces, correspondence, discussions, lessons learned, and expert opinion papers were excluded. This resulted in 257 papers, which were reviewed independently by two researchers. These papers were assessed with regard to quality and relevance to the second SLR. The assessment was based on the following criteria: objective of study, context description, research design, data collection and analysis, and justification of findings. All disagreements were resolved by discussion between the two researchers. Of the evaluated 257 studies, 52 were finally accepted and included as primary studies in the second SLR.

3.1.2 Data Analysis and Reporting

Data synthesis involves combining and summarising the results obtained from the primary studies by using qualitative synthesis. The data was analysed thematically, and themes recurring across the primary studies were found. The analysis of first SLR, presented in Paper I, inspired subsequent studies, along with the baseline for interviews. The analysis of the second SLR is presented in Paper II.

3.2 Phase 2: Empirical Studies

In Phase 2, empirical studies were conducted to explore the use of Kanban in software companies. Data was collected from major Finnish software companies that were actively using and researching Lean and Kanban. This setting allowed for the collection of rich data, while maintaining the flexibility necessary for an exploratory study. Table 6 summarises the following in relation to Phase 2: data collection technique used, number of participants, and number of companies that participated in each study.

Table 6. Phase 2 research questions, papers, and research methods.

Research question	Paper No.	Title	Data collection technique and participants	Number of companies
RQ2.1	III	Usage of Kanban in software companies—an empirical study on motivation, benefits and challenges	21 Questionnaires, 8 Interviews	10
RQ2.2	IV	Transition of software maintenance teams from Scrum to Kanban	Qualitative study, 17 Interviews	2
RQ2.3	V	Portfolio management and Kanban: an empirical investigation with Agile and Lean software companies	Qualitative study, 8 interviews	7

In Phase 2, trends in the use of Kanban were studied quantitatively in companies participating in the *Cloud Software* and the *N4S* programs. These are large, four-year DIMECC programs funded partly by Tekes [7] (the Finnish Funding Agency for Innovation). In these programs, the industrial partners took the initiative to start using Agile, Lean, Kanban, and related approaches and were interested in knowing and learning more about said approaches.

The quantitative study was followed by qualitative studies exploring the implementation of Kanban in software development, software maintenance, and potential use of Kanban for portfolio management in software companies. In this

[7] Tekes is the most important publicly funded expert organisation for financing research, development, and innovation in Finland. https://www.tekes.fi/en/tekes/

regard, semi-structured interviews were conducted with Kanban practitioners from Finnish software companies. The results helped answer RQ2 and its sub research questions (RQ's 2.1–2.3).

3.2.1 Survey

As part of an empirical study, *a survey provides a quantitative or numeric description of some fraction of the population — the sample — through the data collection process of asking questions of people* (Creswell, 1993, Fowler, 2013). Survey, as quantitative research method, is used commonly in both empirical software engineering research and information systems research to elicit data from a variety of sources including individuals, groups, and organisations (Wohlin & Aurum, 2014). To identify the trends in and the current state of the use of Kanban, an Internet survey was conducted in software companies. The results of this survey provided an overview as well as in-depth understanding of the breadth of Kanban.

Data Collection

The questionnaire was administered online using the Webropol tool[8], and it mostly comprised close-ended questions, using both multiple choices and Likert scales for answers. The survey aimed to rate the importance of motivations for using Kanban, and the benefits of and challenges associated with Kanban adoption. In the introduction to the questionnaire, it was clearly mentioned that the target respondents were industrial practitioners with experience of using Kanban in their work.

The data was collected from companies that participated in the *Cloud Software* and the *N4S* programs. The survey was piloted with select representatives from the participant companies and researchers for checking the consistency of the questionnaire. Then the survey was emailed to software companies, and it was kept open for two weeks. A reminder was sent during the second week of the survey.

[8] https://www.webropolsurveys.com/ (accessed June23, 2016).

Data Analysis and Reporting

IBM SPSS Statistics software[9] was used for data analysis. The data analysis was performed using descriptive statistics (only Mean and Median) and organised according to the sections of the survey, alongside a comparison of the results with those of earlier studies on the topic. Additionally, the survey results were complemented with semi-structured interviews with the participants who completed the online survey. Such data triangulation helped minimise ambiguity about the underlying concepts and questions, in addition to helping obtain meaningful answers. The results were reported in paper III.

3.2.2 Interviews

Qualitative research methods originated from sociology and anthropology (Hove & Anda 2005, Seaman 1999, Taylor & Bogdan 1984). In qualitative research, data is frequently collected through interviews to gain insight into the interviewees' worlds, their opinions, experiences, thoughts, and feelings. The interviewer encourages interviewees to freely present their viewpoints, opinions, and experiences. Qualitative research methods help provide a rich picture of the interviewees' viewpoints and deeper insights into the phenomenon under study (Denzin & Lincoln 2005). Typically, qualitative data is collected directly through individual interviews, focus groups, observations, and action research.

In Phase 2, qualitative data was collected through interviews. Interviews can be structured (have close-ended questions), unstructured (have open-ended questions) or semi-structured (have a mix of close- and open-ended questions). To investigate the use of Kanban in software companies, semi-structured interviews were conducted according to the guidelines prescribed by Patton (1990): identification, themes, design, interview, transcription, analysis, and reporting. The interview themes were based on the results of Phase 1 studies and influenced by the partner software companies' inputs.

Selection of Interviewees

The key informant technique was adopted for selecting interviewees (Kumar *et al.* 1993). The technique is good for identifying key experts who can provide rich and

[9] http://www-01.ibm.com/software/analytics/spss/ (accessed June23, 2016).

high-quality data. This technique allowed us to focus on highly knowledgeable individuals and interview the optimum number of people (Kumar *et al.* 1993, Patton 1990). Data was collected from the key informants, who were selected based on the following criteria:

- Interviewees must have experience with participation in projects conducted through Kanban, and
- Interviewees must represent different roles.

The key informants in Paper III were eight managerial-level company representatives. In Paper IV, 17 interviewees from two large Finnish software companies were interviewed. These interviewees had been using Kanban for more than one year for software maintenance work. The key informants in Paper IV were from various positions: product owners, project managers, coaches, Scrum masters, team leaders, developers, testers, and trouble reporters. In paper V, senior executives were considered the most suitable informants because they have knowledge and experience of Agile, Lean, and Kanban, and they participate in their respective companies' portfolio management-related activities.

Interview Technique

Before the actual interviews, each study interview protocol was piloted with one expert from the software industry and three researchers. At least two researchers participated in all interview sessions. At the start of each interview session, the study objectives and the definitions of key terminologies were presented to the interviewees.

Data Analysis and Reporting

All interviews were audio recorded and notes were made. The audio recordings were transcribed and sent back to all interviewees for review and response validation. The interview transcripts were analysed using NVivo 10 software[10] with thematic analysis for detecting themes and patterns in the collected information (King, 1998).

Template analysis is a systematic technique for categorising qualitative data thematically. Template analysis was selected because it allows for the use of *a*

[10] http://www.qsrinternational.com/what-is-nvivo(accessed June23, 2016).

priori themes to help develop an initial version of the coding template. An initial interview template was constructed based on the existing Kanban literature. Interview data was then mapped onto the initial template, modifying it further until all the relevant data was coded. The process was concluded by applying the final version of the template to the data as a whole, and *a priori* themes were redefined or discarded if they did not prove helpful in capturing key details from the data.

4 Original Research Papers

In this section, we present the publications included in the dissertation. The dissertation consists of the following five papers published in peer-reviewed international conferences and journal in the fields of software engineering and information systems:

- IEEE 39th Euromicro Conference on Software Engineering and Advanced Application,
- 9th and 10th International Conferences on Software Engineering Advance,
- IEEE 49th Hawaii International Conference on System Sciences, and
- Journal of Software: Evolution and Process (Wiley).

All publications are ranked in Julkaisufoorumi (JUFO)[11], a rating and classification system created by the Finnish scientific community for quality assessment of academic research.

Table 7 summarises the contributions of each paper to the various research questions. In the following subsections, we elaborate each paper. The author of this dissertation had major involvement in every research phase. The author of this dissertation was the main author of publications I, III, V, VI, and V. The co-authors contributed significantly in each study in terms of data collection, literature review, questionnaire formulation, data analysis, and qualitative data coding. The main author took the lead in writing the manuscript, and the co-authors provided support in structuring it logically and contributed in writing all the papers. Finally, the main author formally presented the results in the abovementioned conferences.

[11] http://www.julkaisufoorumi.fi/

Table 7. Summary of publications in this dissertation.

Publication	RQ	Purpose	Main findings	Author's contribution
I	RQ1.1	To understand Kanban and analyse the trend of its use in software development, along with benefits obtained and challenges faced.	There is growing interest regarding the use of Kanban in software companies. Research on Kanban in software engineering is in the early stages. In the existing studies, Kanban use is described on an abstract level and mainly reported in terms of the use of two Kanban principles: workflow visualisation and WIP limit. The findings show that 84% of the studies on Kanban were related to co-located projects. No study has reported Kanban use in distributed software development projects.	Major involvement in all research phases. Main author of the paper.
II	RQ1.2	To analyse industrial engineering Kanban and compare it with Kanban in software engineering	Kanban in software engineering shares four characteristics with Toyota's Kanban: pulled production, decentralized control, limited WIP, and two types of signals (production and transportation signals). Five other variations of industrial engineering Kanban were identified. The results suggest electronic Kanban for distributed software development.	Major involvement in all research phases. Main author of the paper.

Publication	RQ	Purpose	Main findings	Author's contribution
III	RQ2.1	To investigate the status and trends of Kanban in software companies	Use of Kanban in Finnish software companies is growing. The key motivation factors for adopting Kanban were to improve team communication and development flow, reduce time to market, and create transparency in the company. The results show that Kanban helps improve work visibility, team communication, and workflow control. The common challenge in adopting Kanban is a lack of understanding of the principles of Kanban. Results suggest that Kanban training is important to understand its principles. Furthermore, existing working practices should be respected and Kanban needs to be implemented incrementally in companies.	Major involvement in all research phases. Main author of the paper.
IV	RQ2.2	To investigate the drivers of Kanban use in software maintenance work.	Kanban is more appropriate for work in which there is a high degree of variability in priority. It brings visibility to maintenance tasks and helps in task prioritisation and synchronisation of work with other teams and management.	Major involvement in all research phases. Main author of the paper.
V	RQ2.3	To investigate possible use of Kanban for portfolio management in software companies	There is interest in the use of Kanban for portfolio management. It brings visibility to the workflow and clarity with respect to high-priority activities measured against resources.	Major involvement in all research phases. Main author of the paper.

4.1 Paper I: Kanban in Software Development: A Systematic Literature Review

This paper analysed the trend of Kanban use in software development and identified potential areas for research, as well as knowledge gaps that demand further investigation. Through literature analysis, the study highlighted the use of Kanban, along with its benefits and the challenges in its implementation, as well as its maturity level in software engineering. To the best of the author's knowledge, this study was the first SLR on Kanban in software engineering. Paper I answered RQ1.1.

The search strategy yielded 1828 papers, of which 19 were finally selected as the primary studies. The rest of the papers were excluded because they were not aligned with the main goals of paper I or did not pass the defined minimum quality threshold. Paper I identified a growing number of studies on Kanban in software development, revealing that its use is gaining momentum in the field. The results showed that the Kanban principles of workflow visualisation and WIP limit have been used extensively in software companies. Workflow visualisation helps developers understand the overall direction of work, thereby helping improve team motivation and communication. Kanban helps software development teams in setting WIP limits, which helps with task prioritisation and efficient execution, minimising context switching, and guiding developers to focus on one task at a given time.

The major challenges in using Kanban are people's mindset, lack of understanding of Kanban principles, and difficulty in changing organisational culture. To avoid these challenges, a number of measures were suggested in Paper I, for example, providing Kanban training and implementing it incrementally or using Kanban as a *plug-in* with the existing way of working.

The study revealed that research on Kanban is currently in the early stages. Since 2008, studies on Kanban in software engineering have been increasing each year. The majority (53%) of the primary studies on Kanban in software development were published in 2011. However, the SLR highlighted that the major portion of published research consists of experience reports; and most of the studies on Kanban involved pilot or small-scale software development projects. Additionally, no study has reported the use of Kanban in distributed software development projects. The use of Kanban in various industrial settings or different types of software projects could also be an interesting area for future research.

4.2 Paper II: Kanban in Industrial Engineering and Software Engineering: A Systematic Literature Review

In the 1940s, kanban was first used by Toyota for manufacturing, and in last decade, it was adapted for software engineering. No scientific study discusses Kanban in software engineering in the light of industrial engineering, the domain of origin of Kanban. To the best of our knowledge, Paper II was the first study to investigate Kanban in both industrial engineering and software engineering literature and report on their differences and similarities. Paper II answered RQ1.2.

The search strategy yielded 1552 papers, of which 52 studies[12] were finally selected as the primary studies. The rest of the papers were excluded because they were not aligned with main goals of paper II or did not pass the defined minimum quality threshold. The SLR on Kanban in industrial engineering provides insights about Toyota Kanban and five other variations of Kanban in the literature. In 1977, Sugimori *et al.* published the first academic paper describing kanban, entitled 'Toyota Production System and Kanban System: Materialization of Just-In-Time and Respect-For-Human System'.

The four basic characteristics of Toyota Kanban are pulled production, decentralised control, limited WIP, and two types of signals (i.e., production signals and transportation signals). These characteristics are similar to Kanban in software engineering. Kanban in software engineering and kanban in industrial engineering use the *pull system*. Based on the mentioned characteristics, there are no conceptual differences between Kanban in software engineering and industrial engineering. However, there are practical differences. For instance, Kanban in industrial engineering easily manages physical items with one-piece flow owing to low variation in production lines. Additionally, *in manufacturing, Kanban applies to repetitive work—building the same item again and again* (Liker, 2004). By contrast, in software development Kanban manages non-physical items, which vary in size and complexity, and repetitive building of the same item is very rare.

The other five variations of Kanban in industrial engineering are Generic Kanban, Generalised Kanban Control System, Extended Kanban Control System, Flexible Kanban System, and Electronic Kanban. These variations of Kanban have similar characteristics because they are fundamentally concerned with signals: production signal (authorises a process to produce a fixed amount of product) and

[12] In the original publication there is a typo, the actual primary studies are 52. A complete list of primary studies can be found in Appendix A.

transportation signal (authorises the transport of a fixed amount of product to the next workstation). The differences lie in these signals, for instance, electronic Kanban has one modification—the use of electronic signals in place of physical signals. The results suggest the use of Electronic Kanban for distributed software development because it provides the possibility to visualise workflow for remote teams and obtain up-to-date work status instantaneously. Furthermore, the use of Generic Kanban is suggested for software maintenance because it is effective in environments with unstable demand, as well as in environments with large variability in processing time.

The application of Kanban in industrial engineering and software engineering provides similar benefits. Specifically, Kanban creates a smoother development flow, reduces cycle time, and improves quality. For future research, Paper II called for studies that consider the identified variations of Kanban in practice and evaluate their effectiveness in software companies.

4.3 Paper III: Usage of Kanban in Software Companies—An Empirical Study on Motivation, Benefits and Challenges

Paper III extended the work conducted in Paper I and provided valuable descriptive information about the contemporary state of Kanban use in software companies. This study makes three main contributions. First, it provides up-to-date knowledge on the state of Kanban use in software companies and identifies motivation factors for using Kanban. Second, it identifies the benefits of and challenges faced with the adoption and use of Kanban. Third, it elaborates possible solutions to the identified challenges. Paper III answered RQ2.1.

The participating companies were selected from the *Cloud software* and the *N4S* programs. Data was collected through an online questionnaire and semi-structured thematic interviews with representatives of the selected software companies. The survey was open for two weeks, and 21 responses were received during this time. These respondents represented 10 different large software companies. To complement the survey findings, eight managerial-level company representatives were interviewed. The duration of the interviews varied between 60 and 90 minutes (average 70 minutes).

The survey respondents were mostly from mid-level management (project managers, program managers, agile coaches, and analysts), and 76% had more than 10 years of software development experience. All the respondents had been using Kanban for more than one year, and they rated themselves as being competent in

their knowledge of Kanban. 57% of the survey respondents reported that most of their software development teams are experienced users of Kanban in the company.

The findings highlighted that the main motivation factors for adopting Kanban were improving team communication and development flow, reducing time to reach the market, and creating transparency in the organisation. The findings show that along with software development, the software companies successfully implemented Kanban in testing teams.

The most common benefits of using Kanban were improved transparency of work and communication. The interviewees explained that the use of Kanban helps to make work visible inside and outside the team and better addresses customer needs. As a result, the team members can see the bigger picture of work and select high-priority tasks easily. Furthermore, the findings showed that with Kanban usage, teams work more collaboratively, thus helping getting things done quickly and successfully.

The biggest challenges in using Kanban were lack of experience, lack of Kanban training, and unfavourable organisational culture. These findings highlighted that the lack of experience in using Kanban leads to other challenges, e.g., difficulty managing the WIP limit and selecting tasks according to priority. Further, owing to the lack of training and coaching, the teams misunderstood the purpose and theory behind Kanban. Such misunderstandings become the reasons for teams reverting to their previous way of work.

To tackle such challenges, it is recommended that software companies provide proper Kanban training and coaching and allow teams to experiment in their work. Awareness about Kanban is required from top-level management down through the company. The message of what problem the company is trying to solve with Kanban needs to be communicated. Sudden changes in the way of working should be avoided in a company. Companies should introduce and implement Kanban incrementally without disrupting existing work practices. The results obtained in Paper III were largely in line with the findings of Phase 1.

4.4 Paper IV: Transition of Software Maintenance Teams from Scrum to Kanban

Paper IV investigated how software companies benefit from Kanban in software maintenance work. Software maintenance is closely related to software development because it includes activities such as responding to changes, meeting new requirements, and fixing errors in existing software. To the best of the authors'

knowledge, this was the first peer-reviewed study to investigate the use of Kanban in software maintenance work. Paper IV answered RQ2.2.

Kanban implementation was explored in two software maintenance teams from two large Finnish software companies. In this study, 17 interviews (nine face-to-face interviews and eight via Skype) were conducted, each lasting 1–2 hours. From each team, product owners, project managers, coaches, Scrum masters, team leaders, developers, testers, and trouble reporters were interviewed. Thematic analysis was applied to identify the most common challenges in the previous way of working (i.e., Scrum) and their solutions in Kanban.

Before applying Kanban, these teams were facing challenges in software maintenance work, including lack of work visibility, task prioritisation, communication, and collaboration, in addition to problems with work synchronisation and moving people or domain experts from one team to another. The teams and their change management departments realised that their current working practices needed attention to address these challenges. The management and the teams decided to introduce Kanban in software maintenance work. Both teams received one week of Kanban training. The teams used Kanban on an experimental basis. After the trial period, both teams decided to continue with Kanban in their work.

The results showed that by providing proper training and allowing the teams to experiment with Kanban in their own work contexts, the challenges encountered by them were mitigated efficiently. Additionally, the findings highlighted that Kanban can be optimised to work situations where task flow changes unpredictably, as is the case in software maintenance work. The benefits accrued by the software maintenance teams from using Kanban were increase in individual task visibility, improvement of team morale, and increased knowledge sharing. The obtained results were in line with the findings of Phase 1. For future research, it is recommended that this study be augmented by performing additional studies with other teams using Kanban in software development and maintenance.

4.5 Paper V: Portfolio Management and Kanban: An Empirical Investigation with Agile and Lean Software Companies

Kanban practitioners and educators promoting Kanban argue that it can be used as a tool for managing software projects, products, services, and other offerings in companies' portfolios. However, the literature lacks empirical evidence to back this

suggestion. In this regard, Paper V explored the role of Kanban in portfolio management in software companies. Paper V answered RQ2.3.

Data was collected through face-to-face, semi-structured interviews with key informants from seven software companies participating in the *N4S* program. In total, seven interviews were conducted, with one interviewee from each company, except for one company that had two key informants. The eight interviews resulted in a total of 582 minutes of recorded material and 50 pages of interview transcripts. The interviews were analysed against four goals of portfolio management: 1) tools and methods for maximising portfolio value, 2) tools and methods for ensuring balance in the portfolio, 3) tools and methods for ensuring that the portfolio reflects business strategy, and 4) tools and methods for ensuring a proper number of projects against organisational capacity.

These software companies use Kanban for portfolio management, but the practice is in its early stages. Similar to software development and software maintenance, Kanban in portfolio management brought visibility to the company's portfolio and helped identify the most important offerings and high-priority activities, as well as improve resource allocation. This visibility helped the software companies to obtain immediate feedback regarding their offerings from various stakeholders, which was useful from the viewpoint of portfolio value maximisation. Furthermore, software companies gained from implementing Kanban by limiting the number of ongoing offerings in the portfolio and working to align the offerings with company strategies. The results showed that the applicability of Kanban is not limited to software development or maintenance; instead, it can also be applied to portfolio management in software companies. The results would be helpful for companies planning to implement Kanban for portfolio management. For future research, Paper V suggested comprehensive studies on the use of Kanban for portfolio management in both large and small software companies.

5 Discussion and Conclusion

The introduction of Kanban in software engineering has proved to be quite successful. The body of knowledge on Kanban in software engineering is dominated by consultants' interpretations and claims in the context of software companies. Undoubtedly, consultants make valuable contributions, but the scientific body of knowledge on Kanban in software engineering is lacking. To obtain better understanding of Kanban, this dissertation explored Kanban in software engineering and industrial engineering.

5.1 Answer to RQ1: What is the understanding of Kanban in software engineering based on literature?

Kanban in software engineering is similar to Toyota's Kanban. It inherits four basic characteristics: pulled production, decentralised control, limited WIP, and two types of signals (i.e., production signals and transportation signals) (see Paper II). Kanban in both software engineering and industrial engineering uses a card to visualise each work item and signal the current work. WIP limits control the work, and do not allow individuals to exceed system or team capacity. In software development, WIP limits, continuous flow, and the pull system can be achieved by using Kanban. Kanban focuses more on enabling task visualisation and self-direction, so as to help team members become autonomous and improve their own processes. Kanban in both industrial engineering and software engineering yields benefits such as smoothing development flow, bringing visibility to the task status, reducing cycle time, and improving quality.

In software engineering, the majority of the studies on Kanban have been published since the year 2011 (see Paper I). The existing studies have used two main principles of Kanban: workflow visualisation and WIP limit. Value stream mapping is a popular method for visualising workflow. Task visualisation makes it easier for the software development team to understand the overall workflow.

Kanban adoption in software companies and the accompanying research are still in the early stages. Before 2013, no study elaborated how Kanban is used or could be used in software development. The growing interest in using Kanban in software companies has resulted from the focus of Kanban on development flow, improving teams' internal and external communication, and promotion of the pull system. This growing interest is continuance and in recent years more studies are reporting wider application of Kanban in software companies (Al-Baik & Miller

2016, Concas et al. 2013, Kerzazi & Robillard 2013, Laanti & Kangas 2015, Neely & Stolt 2013, Sjøberg et al. 2012, Tripathi et al. 2015, Turner et al. 2013, Turner 2014). Further, some universities started to include Kanban introduction in software engineering degree programmes curriculum (Ahmad *et al.* 2014a, Ahmad *et al.* 2014b, Heikkilä *et al.* 2016, Liskin *et al.* 2014, Scharlau 2013, Oza *et al*. 2013).

It is important to emphasize the understanding and visualisation of current works, development processes, and related activities. Collective understanding of workflow motivates the entire team to work collaboratively and control bottlenecks in their work. For instance, when a task is blocking the workflow, software development team members help each other to complete the task and move forward. Workflow visualisation helps developers prioritise tasks efficiently and improves communication inside and outside of software development teams. Furthermore, transparency empowers development teams to interact openly with the management, and once an issue emerges in the process of software development, it can be resolved collectively. This aspect is important for achieving continuous improvement (Kaizen).

Following visualisation and understanding of the current workflow, Kanban ensures the minimisation of WIP in each development stage. The WIP limits help software development team members to minimise context switching and focus on a single task at a given time. Once a task is assigned to a team member, the member should focus on completing it as soon as possible. Thus, individual team members handle and control their own tasks better, which leads to improved quality of work and avoidance of blockage in the workflow. In addition, WIP limits enable development teams and other stakeholders to keep track of and distinguish completed and WIP tasks.

It is recommended that the WIP limit should be implemented with consensus between the development team and other related stakeholders. It is important to limit WIP tasks based on team capacity, as opposed to some arbitrary schedules. Once a team sets a WIP limit for its work, it will automatically find a way to use the pull system—start pulling high-priority tasks from the backlog. In this way, WIP limits and the pull system help ensure smooth flow of development work. *It is recommended for the teams when using Kanban to constantly Plan-Do-Check-Act in order to enable Kaizen. Furthermore, pay attention to the work that is flowing from backlog till completion.* In software engineering, the use of various instruments to measure flow has been reported, for example, cumulative flow diagrams, lead time/cycle time, and defect rate.

Despite the benefits of using Kanban, software companies face a number of challenges such as lack of experience in using Kanban, misunderstanding the Kanban process, resisting change, and failing to adapt to changes in organisational culture. Another hidden challenge is that software companies violate the *incremental change rule* in the adoption of Kanban. For example, a software company claimed that their *process transition* from Scrum to Kanban was made overnight (Nikitina & Kajko-Mattsson 2011).

When introducing Kanban, it is recommended to not change the existing team working practices or processes. To efficiently implement and adopt Kanban, it is important to visualise the flow of value in the organisation while respecting the existing process, roles, responsibilities, and titles. Kanban adoption requires organisational support and time to become effective. Incremental change along with organisational support will help diminish the resistance to the use of Kanban.

5.2 Answer to RQ2: How is Kanban used in software companies?

Kanban's situational adaptation makes it a useful process management tool for all types of work in software companies, including software development, software maintenance, and portfolio management. The highest motivation factors in Kanban adoption are to improve team communication, reduce time-to-market, improve development flow and create transparency within the companies. Kanban acts as a change method with a feedback mechanism catalysed by a pull system. Software companies have used Kanban in software development, as elaborated by Anderson (2010), and have achieved various benefits such as improved visibility and transparency of work, improved communication, and better control of workflow and WIP (see Paper III). These benefits are aligned and confirmed with the findings of the literature review (see Paper I) and consultants claims.

The major challenges faced by software companies in the use of Kanban in software development were lack of experience, misunderstanding the process, resisting change, and failing to adapt to changes in organisational culture. Paper III and IV suggested that, *adopting Kanban requires organisational support and time to become effective.* In this regard, software companies should focus on *double-loop learning* instead of *single-loop learning*. Argyris & Schön (1996) distinguished between single- and double-loop learning in organisations. Single-loop learning involves changing practice as problems arise to avoid the same problem in the future. By contrast, double-loop learning involves challenging and ultimately changing the underlying organisational culture (Argyris & Schön 1996,

Al-Baik & Miller 2016). Single-loop learning is about asking *are we doing things right*? Double-loop learning is about asking *are we doing the right things* (Moe 2013). Software companies must create an environment in which people can explore their underlying assumptions and compare the results of change actions with the actual outcomes.

Software companies should provide freedom to individuals to control their tasks and encourage collaborative work. In this way, not only the individuals but also the entire company will become more skilled. *It is recommended to provide Kanban training and allow teams to experiment or pilot Kanban in their work.* By providing training, coaching, and allowing reasonable time for teams to pilot or experiment with Kanban in their work, software companies can greatly increase the speed and success of Kanban adoption.

Software companies are also taking advantage of using Kanban in software maintenance (see Paper IV). Kanban can be used in situations where tasks change frequently and unpredictably, for example, software maintenance. Software maintenance work includes dealing with customers' requests, related mainly to improvements in the existing product or the incorporation of new features. The participating companies provided Kanban training to software maintenance teams and gave them sufficient time to experiment with it in their work. The establishment of such a supportive environment helped individuals share their problems and openly debate how to solve them. In addition, it promoted a learning culture in the company. Such a culture helps people evolve gradually based on their experiences by encouraging learning from failure.

With Kanban implementation, software maintenance teams mitigated challenges such as the lack of work visibility, task prioritisation, communication and collaboration, work synchronisation, and over-commitment. Furthermore, the companies themselves experienced benefits such as increased visibility of tasks, improvement of team morale, and better communication and knowledge sharing. Dealing with legacy code required the movement of experts from one area to another. Using Kanban made it easier to move experts or resources from one team to another. By using Kanban in software maintenance, the teams' actions in response to urgent work were more spontaneous, and work could be pulled according to priority. WIP limits improved software maintenance teams' throughput and efficiency. The results suggest that Kanban is an evolutionary method that can be optimised to a working situation in which tasks change frequently, for instance, software maintenance.

Kanban implementation in software maintenance teams is an example of how an organisation can apply the double-loop learning theory in practice (see Paper IV). The companies provided an environment (e.g., Kanban board) where information was shared openly and was available readily. The visualisation of tasks allowed teams to explore the underlying assumptions openly. During the trial or experimental period of Kanban use in their work, the team members jointly protected each other in learning and risk taking. The teams were looking to understand the system. The team members were involved in prioritisation of tasks, defining policies, and moving tasks on the board upon completion. The teams adopted Kanban in practice and enjoyed the freedom of individual pull, controlling their own tasks, and focusing on collaboration when needed. Such an environment helped ensure that the attempted improvements were achieved and sustained.

The growing interest in the use of Kanban in software companies is not limited to software development and software maintenance. Various consultants have claimed that Kanban is a good tool for higher management from the viewpoint of managing software projects and services in their company portfolios. Paper V confirmed and discusses that software companies have taken interest in the use of Kanban for portfolio management. The participating companies claimed Kanban helped provide visibility to software projects and services in the company's portfolio. Paper V provided a deeper understanding of Kanban implementation for portfolio management in software companies. *The software companies were using Kanban in a similar fashion to that explained by Leffingwell (2010).* The companies experimented with a popular Kanban tool (i.e., the JIRA[13] Kanban board) and created a separate board for each of the following:

- Business-line Kanban, containing the releases of all products and services. Here, business goals were linked using Kanban, which made visible the investment in each product or service line. This helped the portfolio managers to visualize the company's activities holistically.
- Program-level Kanban, containing the features of a specific product or service. This level served as a bridge between business lines and lower teams.
- Team-level Kanban, containing user stories related to a specific product or service. Here, the actual development of the product and services took place.

The companies' portfolio Kanban board visualised offerings across all business units and teams. The portfolio Kanban boards helped users view and understand

[13] https://www.atlassian.com/software/jira

the progress of their respective companies' offerings, which were in various stages (e.g., development, testing, in market). Additionally, the companies emphasised the importance of getting immediate feedback from customers and the managers of the various product and services in terms of the immediate delivery of offerings from the viewpoint of portfolio value maximisation. In such situations, Kanban triggers the need for communication and coordination within teams or business units. Kanban helped the management representatives to identify high-priority projects and services, and it brought their attention to the allocation of resources where needed. In other words, Kanban helped the management representatives to balance demand against capacity. Kanban did not balance the portfolio, but it provided signals (for example, that the working pipeline is full) and, in turn, helped the management to evaluate suitable available options at a given time and take necessary actions.

5.3 Threats to Validity

Validity represents the trustworthiness of research results. It is important to consider threats to validity during study design to increase the validity of the findings. In this dissertation, threats to validity were considered throughout the research process by following the guidelines outlined by Runeson & Höst (2009) and Yin (2009). The four aspects of validity are: construct validity, internal validity, external validity, and reliability.

Construct validity reflects the extent to which the studied operational measures really represent what the researcher had in mind and what is investigated according to the research questions (Runeson & Höst 2009). For example, there is a risk that questions can be misunderstood or not interpreted in the same way by the researcher and the respondents. Therefore, the survey (Paper III) and interviews (Papers III, IV and V) were pre-tested and piloted internally with researchers and collaborating experts from the selected software companies. At the beginning of each interview session, the interviewees were introduced to key concepts and terminologies used in the study in order to avoid misinterpretation. An additional potential threat is the selection of interviewees with a view to obtaining appropriate data in order to answer the research questions. Therefore, the key informant technique was applied for selecting the pool of interviewees.

Reliability is concerned with the extent to which data collection and analysis are dependent on specific researchers (Runeson & Höst 2009). The risk is that researcher's preconceptions in data collection and analysis may affect the reliability

of a study. To improve reliability and reduce the risk of researcher's bias, investigator triangulation was applied in data collection and analysis. The key informants' statements minimised researcher's bias in the data collection process; however, the data collected from key informants may be affected by their own subjective opinions and knowledge.

Internal validity is a concern when examining causal relations (Runeson & Höst 2009). According to Yin (2009), internal validity is primarily related to explanatory studies 'when an investigator is trying to explain how and why event x led to event y'. In exploratory research, as conducted in this dissertation, internal validity is concerned with the inferences made when establishing findings and drawing conclusions from empirical evidence. Therefore, data triangulation was performed to ensure internal validity; different types of studies were conducted, both qualitative and quantitative data was collected, and participants had different profiles in the studies. In paper IV, data is only collected through single source 'interviews'. However, to mitigate this threat data was collected from two different companies in which different roles of the participants were taken into consideration.

External validity is concerned with the extent to which findings are generalizable and are of interest to other people outside the investigated case (Runeson & Höst 2009). Often, external validity depends on the nature of sampling used in a study. The studies reported in this dissertation were conducted in the Finnish software companies selected from the Cloud Software and the N4S programs. These companies have branches in European countries, the United States, and Asia, and they are active users of the Lean and Kanban. The participating companies differed in terms of size and nature of business, which might have affected the results, as might have other factors such as company culture and application area. Finnish software companies cannot be assumed to represent Kanban software companies in general. Although Finnish software companies are well-known for their efficient use of Agile and Lean (Bilbao-Osorio *et al.* 2013, Rodríguez 2013), further study is required to help generalise the results.

5.4 Summary of Contributions

The synthesis of the studies conducted in this dissertation are summarised in the following four main contribution points.

1. *There is a growing trend of Kanban use in software companies.* Based on the studies and latest literature, it can be seen that in software engineering, the use

of Kanban was started in software development, and its implementation is being extended to software maintenance and portfolio management. Software companies are extensively using two principles of Kanban, (1) visualisation of work and (2) limited WIP, to achieve constant flow and quick delivery to end users. The results of the studies confirm that visualising and sharing work progress openly across all levels enhances transparency and that teams obtain the bigger picture of their work and make collective decisions easily. For instance, the WIP limit number is selected through a consensus between the development team and related stakeholders.

2. *Kanban training and experimentation help to enable double-loop learning.* The results of the studies show that the successful adoption of Kanban requires proper Kanban training and management support along with a combination of internal and external change agents, mentors, and coaches. The studies confirmed that when people are educated and the expected benefits of Kanban are communicated, they will more likely be convinced to adopt Kanban in their work. The study results also show that it is important for management to understand that the adoption of Kanban in various teams requires experimentation with the help of internal and external change agents or coaches. The results also show that such experimentation creates an environment of continuous learning.*Avoid big bang Kanban transition and blend Kanban with other agile methods.* All the studies conducted in this dissertation recommended that when introducing Kanban, no drastic changes should be made in the workplace. The focus is to evolve work processes incrementally to eliminate the initial fear of Kanban use and avoid resistance from inside the teams or from the organisational culture. The results of the studies conducted in this dissertation show that Kanban was typically used in combination with Scrum in software companies.
3. *The proactive role of team leaders and higher management is essential in Kanban use.* For instance, the study results show that the team leader should help team members to prioritise work based on its severity or urgency and encourage a pull system. This practice enables team members to autonomously pull a task from the Kanban board as their capacity frees up and removes the option of managers pushing tasks to individuals.

5.4.1 Implications for Practice

This dissertation provides insight into Kanban implementation in software companies. A strong practitioner-driven movement and the results of this dissertation support the idea of using Kanban in software engineering. Kanban is a good way to execute the Lean principle and obtain results in software companies that are similar to those produced in industrial engineering. The results of this doctoral dissertation will be useful for software companies from the viewpoint of understanding how peers are using Kanban in software development, software maintenance, and portfolio management.

The dissertation shows that visualisation and transparency are important in software product or services development, where many tasks run in parallel. In this regard, Kanban provides visibility to tasks and workflow, stresses the importance of changing from a push to a pull system, and limits the number of simultaneous WIP tasks, which helps in waste prevention.

The simplicity of Kanban facilitates quick situational adaptation, but many people in software companies require Kanban training to become familiar with the concept. For example, Paper IV showed that providing basic Kanban training facilitated fast adoption and the achievement of target outcomes. Individuals in an organisation must understand that Kanban is a continuous process improvement journey, and each company needs to explore what works best in its specific context.

Kanban adoption implies deep changes in company culture and people's mindsets. An abrupt transition to a new way of working often leads to failure. It is recommended that software companies use the double-loop learning theory (Argyris & Schön 1996) when introducing a new way of working such as Kanban. Kanban encourages teams to constantly Plan-Do-Check-Act in repeated cycles, so that they can learn from their mistakes and move towards continuous improvement. Such learning results in organisational behaviour that enhances the underlying learning. Additionally, the double-loop learning theory helps with parallel optimisation of current practices and experimentation with new ones. To optimise the advantages of Kanban use, software companies should view it as an ongoing and continuous process of improvement.

5.4.2 Implications for Research

Kanban has received considerable attention in the software industry. However, the body of research on Kanban is still in its infancy, and interpreting Kanban in

software engineering is an emerging topic. This research shows a clear need for more empirical studies of Kanban in software engineering. Kanban has five different variations in industrial engineering, while in software engineering, Toyota Kanban is adapted. This dissertation has contributed to improving the understanding of the use of Kanban, as well as the associated benefits and challenges, in various contexts in software companies, namely, software development, software maintenance, and portfolio management. This is a new research area where significant insights can be gained.

Kanban is easy to understand, but the concept is broad. Owing to domain-specific differences, different interpretations are necessary in software engineering more than in industrial engineering. For instance, in software development, different intangible products are created each time, whereas in manufacturing, the same tangible products are made repeatedly.

5.5 Recommendations for Further Research

Despite the growing popularity of Kanban use in software companies, the existing research on Kanban is very limited. Limited scientific support is available to software companies interested in adopting Kanban. Replicating or conducting more extensive studies similar to those conducted in this dissertation will be helpful to confirm or refute the findings reported herein. Using various research methods such as direct observation and experimentation with Kanban practitioners will help deepen understanding of this concept.

This dissertation identified various challenges in Kanban implementation, each of which is a candidate to be studied in greater detail. It would be interesting to conduct experiments on Kanban use in software development in multicultural distributed environments. Large-scale software development is difficult owing to a number of dependencies faced by software development teams. Technical dependencies among software components create social dependencies among the collaborative teams, which result in continuous efforts in terms of communication. It would be useful to investigate how Kanban supports collaborative software development by visualising socio-technical dependencies in software companies. Additionally, it would be worthy to explore the human aspects of the Kanban transformation process, such as participants' perceptions of the change process, impediments to change, and acceleration of change.

The existing limited literature has explored the dynamics of Kanban, albeit with a tendency to focus on the obtained benefits or, to a lesser extent, the pitfalls of Kanban

implementation in brownfield software development projects. Brownfield development could be one developing and deploying new software feature or systems in the existing legacy software applications or systems (Boehm 2009).Furthermore, no study has been reported on the use of Kanban in greenfield software development projects (Ahmad *et al.* 2016 a). A Greenfield project could be one developing a system for a totally new environment, without legacy systems (Boehm 2009). The researchers of this study plan to investigate the hidden pitfalls of Kanban in greenfield and brownfield software development projects to discover the reasons underlying the failure of Kanban. An additional goal is to shed light on the phenomenon by discussing similar experiences among industry experts and uncovering topics that are the most challenging for software companies.

References

Abrahamsson P, Conboy K & Wang X (2009) 'Lots done, more to do': the current state of Agile systems development research. European Journal of Information Systems 18(4): 281–284.

Abrahamsson P, Salo O, Ronkainen J & Warsta J (2002) Agile software development methods: review and analysis. VTT Publications 478.

Abrahamsson P, Warsta J, Siponen MT & Ronkainen J (2003) New directions on Agile methods: a comparative analysis. IEEE Proceedings of 25th International Conference on Software Engineering: 244–254.

Agarwal A, Shankar R, Tiwari MK (2006) Modeling the metrics of lean, agile and leagile supply chain: An ANP-based approach. European Journal of Operational Research 173(1):211–225.

Al-Baik O & Miller J (2014) The Kanban approach, between agility and leanness: a systematic review. International Journal of Empirical Software Engineering: 1–37.

Al-Baik O & Miller J (2016). Kaizen Cookbook: The Success Recipe for Continuous Learning and Improvements. IEEE 49th Hawaii International Conference on System Sciences: 5388–5397.

Ambler SW (2009) Scaling Agile software development through Lean governance. Proceedings of IEEE Computer Society ICSE Workshop on Software Development Governance: 1–2.

Anders, D. (2004) Agile management for software engineering: Applying the theory of constraints for business results. Upper Saddle River, New Jersey, Prentice Hall.

Anderson D (2003) Agile management for software engineering: applying the theory of constraints for business results. Upper Saddle River, New Jersey, Prentice Hall.

Anderson D (2010) Kanban: Successful Evolutionary Change for Your Technology Business. Sequim, Washington Blue Hole Press.

Anderson D & Roock A (2011) An Agile evolution: why Kanban is catching on in Germany and around the world. Cutter IT Journal. 24: 6–17.

Anderson D, Concas G, Lunesu MI, & Marchesi M (2011) Studying lean-kanban approach using software process simulation. International Conference on Agile Software Development, Springer Berlin Heidelberg: 12–26.

Andrezak, M., & Schiffer, B. (2010). Kanban and Technical Excellence or: Why Daily Releases Are a Great Objective to Meet. Proceedings of Lean Enterprise Software and Systems. Springer Berlin Heidelberg: 115–117.

Agile Manifesto (2001) Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningha, W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin RC, Mellor S, Schwaber K, Sutherland J & Thomas D Manifesto for Agile Software Development. URI: http://www.agilemanifesto.org/. Cited 2016/06/06.

Ahmad, MO, Liukkunen K, & Markkula J (2014a) Student perceptions and attitudes towards the software factory as a learning environment. Proceedings of IEEE Global Engineering Education Conference: 422–428.

Ahmad MO, Markkula J, & Oivo M (2016a) Pitfalls of Kanban in Brownfield and Greenfield Software Development Projects. International Conference on Agile Software Development: 296–299.

Ahmad MO, Markkula J, & Oivo M (2016b). Insights into the perceived benefits of Kanban in software companies: Practitioners' views. International Conference on Agile Software Development: 156–168.

Ahmad MO, Markkula J, & Oivo M (2014b) Kanban for software engineering teaching in Software Factory learning environment. World Transactions on Engineering and Technology Education (WIETE) 12(3): 338–343.

Argyris, C., Schön, D.A (1996) On Organizational Learning II: Theory, Method and Practise. Reading, Mass, Indianapolis IN, Addison Wesley.

Babar MA & Zhang H (2009) Systematic literature reviews in software engineering: preliminary results from interviews with researchers. Proceedings of the 3rd IEEE International Symposium on Empirical Software Engineering and Measurement. Computer Society: 346–355.

Basili VR (1993) The experimental paradigm in software engineering. Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions: 3–12.

Becker M, & Szczerbicka H (1998) Modeling and optimization of Kanban controlled manufacturing systems with GSPN including QN. IEEE International Conference on Systems, Man, and Cybernetics.1: 570-575.

Bilbao-Osorio B, Dutta S & Lanvin B (2013) The global information technology report 2013. Growth and jobs in a hyper-connected world. Wold economic forum. URI: http://www.weforum.org/reports/global-information-technology-report-2013. Cited 2016/06/29.

Bhim S, Garg SK, Sharma SK, Grewal C (2010) Lean implementation and its benefits to production industry, International Journal of Lean Six Sigma. 1(2): 157-168.

Burrows M, & Hohmann L (2014) Kanban from the Inside: Understand the Kanban Method, connect it to what you already know, introduce it with impact. Chicago, Blue Hole Press.

Boeg J (2012) Priming Kanban: A 10 step guide to optimizing flow in your software delivery system. 2nd edition. Aarhus Trifork, Chronografisk Margrethepladsen A/S Copenhagen.

Boehm B, Lane J, Koolmanojwong S & Turner R (2014) The incremental commitment spiral model: principles and practices for successful systems and software. Crawfordsville, IN, Addison-Wesley Professional.

Boehm, B. (2009). Applying the Incremental Commitment Model to Brownfield Systems Development. Proceedings of 7th Annual Conference on Systems Engineering Research.

Brewer J & Hunter A (2006) Foundations of multimethod research: synthesizing styles. California, Sage Publications, Inc.

Cawley O, Wang X & Richardson I (2013) Lean software development–what exactly are we talking about? In: Lean enterprise software and systems. Berlin Heidelberg, Springer: 16–31.

Chai L (2008) E-based inter-enterprise supply chain Kanban for demand and order fulfilment management. IEEE International Conference on Emerging Technologies and Factory Automation: 33–35.

Cloud Software Program (2010) URI: http://www.cloudsoftwareprogram.org/. Cited 2016/06/ 25.

Cocco L, Mannaro K, Concas G & Marchesi M (2011) Simulating Kanban and Scrum vs. Waterfall with system dynamics. IEEE proceedings of Agile processes in software engineering and extreme programming: 117–131.

Collis J & Hussey R (2009) Business research: A Practical Guide for Undergraduate and Postgraduate Students 3rd Edition. New York, Palgrave MacMillan.

Conboy K (2009) Agility from first principles: reconstructing the concept of agility in information systems development. Information Systems Research 20(3): 329–354.

Concas G, Lunesu MI, Marchesi M & Zhang H (2013) Simulation of software maintenance process, with and without a work-in-process limit. Journal of Software: Evolution and Process 25(12): 1225–1248.

Creswell JW (2009) Research design: Qualitative, quantitative, and mixed methods approaches. 3rd ed. California, SAGE Publications, Incorporated.

Creswell (1993) Research Design – Qualitative and quantitative approaches. Thousand Oaks, Sage Publications.

Cutter (2011) The Viral Growth of Kanban in the Enterprise, The journal of information technology management. 3-29. URI: https://leankit.com/blog/wp-content/uploads/2011/12/Cutter_on_Kanban_from_LeanKit_Kanban.pdf. Cited 2016/07/03.

Dingsøyr T, Dybå T & Abrahamsson P (2008) A preliminary roadmap for empirical research on agile software development. IEEE proceedings of Agile Conference: 83–94.

Dingsøyr T, Nerur S, Balijepally V & Moe NB (2012) A decade of agile methodologies: towards explaining agile software development. Journal of Systems and Software 85(6): 1213–1221.

Dybå T & Dingsøyr T (2008) Empirical studies of agile software development: a systematic review. Information and Software Technology 50(9): 833–859.

Dyba T & Dingsoyr T (2009) What do we know about Agile software development? IEEE Software, 6–9.

Dybå T, Kitchenham BA & Jorgensen M (2005) Evidence-based software engineering for practitioners. IEEE Software, 22(1): 58–65.

Denzin NK & Lincoln YS (2005) Handbook of qualitative research. 3rd Edition. Thousands Oaks, Sage Publications.

Easterbrook S, Singer J, Storey MA & Damian D (2008) Selecting empirical methods for software engineering research. In: Shull F, Singer J & Sjøberg DIK (eds) Guide to advanced empirical software engineering. London, Springer: 285–311.

Ebert C, Abrahamsson P & Oza N (2012) Lean software development. IEEE Software 5: 22–25.

Fagerholm F, Oza N, & Münch J (2013) A platform for teaching applied distributed software development: The ongoing journey of the Helsinki software factory. IEEE 3rd International Workshop Collaborative Teaching of Globally Distributed Software Development: 1-5.

Fowler Jr, FJ (2013). Survey research methods. Fifth Edition. California Sage publications.

Gregory P, Barroca L, Sharp H, Deshpande A & Taylor K (2016) The challenges that challenge: engaging with Agile practitioners' concerns. Journal of Information and Software Technology 77: 92–104.

Gross JM & McInnis KR (2003) Kanban made simple: demystifying and applying Toyota's legendary manufacturing process. New York, AMACOM.

Gravel M, & Price WL (1988) Using the Kanban in a job shop environment. The International Journal of Production Research, 26(6): 1105–1118.

Greaves K (2011). Taming the Customer Support Queue. IEEE proceedings of Agile Conference: 54–160.

Hallam CR (2003) Lean enterprise self-assessment as a leading indicator for accelerating transformation in the aerospace industry, Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA. URI: http://dspace.mit.edu/handle/1721.1/29216 . Cited 2016/07/06.

Heeager LT & Rose J (2015) Optimising Agile development practices for the maintenance operation: nine heuristics. International Journal of Empirical Software Engineering 20(6): 1762–1784.

Heikkilä VT, Paasivaara M, & Lassenius C (2016) Teaching university students Kanban with a collaborative board game. In Proceedings of the 38th International Conference on Software Engineering Companion: 471–480.

Hibbs C, Jewett S & Sullivan M (2009) The art of Lean software development: a practical and incremental approach. 1st ed. Sebastopol, CA, O'Reilly Media, Inc.

Highsmith J (2002) Agile software development ecosystems. The agile software development series. Boston MA, Addison-Wesley.

Hiranabe K (2008) Kanban applied to software development: from Agile to Lean. URL: http://www.infoq.com/articles/hiranabe-lean-agile-Kanban. Cited 2016/04/08.

Hove SE & Anda B (2005) Experiences from conducting semi-structured interviews in empirical software engineering research. Proceeding of 11th IEEE International Software Metrics Symposium: 10–23.

Höst M, Regnell B, och Dag JN, Nedstam J, Nyberg C (2001) Exploring bottlenecks in market-driven requirements management processes with discrete event simulation. Journal of Systems and Software 59(3): 323–332.

Huang CC & Kusiak A (1996) Overview of Kanban systems. International Journal of Computer Integrated Manufacturing 9(3): 169–189.

Hurtado J (2013) Open Kanban - an open source, ultra-light, Agile and Lean method. URI: http://www.agilelion.com/agile-kanban-cafe/open-kanban. Cited 2016/06/26.

Ikonen M, Pirinen E, Fagerholm F, Kettunen P & Abrahamsson P (2011) On the impact of Kanban on software project work: an empirical case study investigation. Proceedings of 16th IEEE International Conference on Engineering of Complex Computer Systems: 305–314.

Ikonen M (2011) Lean thinking in software development: Impacts of kanban on projects. PhD Thesis, https://helda.helsinki.fi/handle/10138/28453. Cited 2016/04/04.

Ikonen M, Kettunen P, Oza N, & Abrahamsson P (2010 a) Exploring the sources of waste in kanban software development projects. Proceedings of IEEE 36th EUROMICRO Conference on Software Engineering and Advanced Applications: 376–381.

Ikonen M (2010 b) Leadership in Kanban software development projects: A quasi-controlled experiment. Lean Enterprise Software and Systems. Berlin Heidelberg, Springer: 85–98.

Kalliney M (2009) Transitioning from Agile development to enterprise product management agility. Proceedings of IEEE International Agile conference: 209–213.

Kettunen P & Laanti M (2008) Combining agile software projects and large-scale organizational agility. Journal of Software Process: Improvement and Practice - Special Issue on Systems Interoperability 13(2): 183–193.

Kerzazi N, & Robillard PN (2013) Kanbanize the release engineering process. Proceedings of the IEEE 1st International Workshop on Release Engineering: 9–12.

King N (1998) Template analysis. In: Symon G & Cassell C. (ed.) Qualitative methods and analysis in organizational research: a practical guide. Thousand Oaks CA, Sage Publications Ltd.

Kitchenham B (2004) Procedures for undertaking systematic reviews. Technical report, Keele University and National ICT Australia. URI: http://csnotes.upm.edu.my/kelasmaya/pgkm20910.nsf/0/715071a8011d4c2f482577a700386d3a/$FILE/10.1.1.122.3308[1].pdf Cited 2016/06/04.

Kitchenham B & Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report, EBSE-2007-01. URI: http://s3.amazonaws.com/academia.edu.documents/35830450/2_14346538958874215 1.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1467614023&Signature=tBZk0gvVA%2FUPpBLwqVoQtycCtGQ%3D&response-content-disposition=inline%3B%20filename%3DSystematic_Literature_Reviews_SLR.pdf Cited 2016/06/04.

Kim CS, Spahlinger DA, Kin JM, Coffey RJ, & Billi JE (2009) Implementation of lean thinking: one health system's journey. The Joint Commission Journal on Quality and Patient Safety 35(8): 406–413.

Kniberg H & Skarinm M (2010) Kanban and Scrum-making the most of both. Enterprise software development series C4Media, Publisher of InfoQ.com.

Kniberg H (2011) Lean from the trenches: Managing large-scale projects with Kanban Raleigh, NC, Pragmatic Bookshelf.

Krafcik JF (1988) The triumph of the Lean production system. MIT Sloan Management Review 30(1): 41–52.

Kumar CS & Panneerselvam R (2007) Literature review of JIT-KANBAN system. International Journal of Advanced Manufacturing Technology 32(3-4): 393–408.

Kumar N, Stern LW & Anderson JC (1993) Conducting interorganizational research using key informants. Academy of Management Journal 36 (6): 1633–1651.

Laanti M (2008) Implementing program model with Agile principles in a large software development organization. 32nd Annual IEEE International Computer Software and Applications Conference: 1383–1391.

Laanti M & Kangas M (2015) Is Agile portfolio management following the principles of large-scale Agile? Case study in Finnish Broadcasting Company Yle. Proceeding of IEEE Agile conference: 92–96.

Laanti M, Similä J & Abrahamsson P (2013) Definitions of Agile software development and agility. European Conference on Software Process Improvement. Berlin Heidelberg, Springer: 247–258.

Larman C (2003) Agile and iterative development: a manager's guide. Boston, MA Addison-Wesley Professional.

Larman C & Vodde B (2008) Scaling Lean & Agile development: thinking and organisational tools for large-scale Scrum. Boston, MA Addison-Wesley Professional.

Leffingwell D (2010) Agile software requirements: Lean requirements practices for teams, programs, and the enterprise. Boston, MA Addison-Wesley Professional.

Leffingwell D & Reinertsen D (2011) Agile software requirements. Upper Saddle River, Boston, MA Addison-Wesley.

Liker JK (2004) The Toyota way: 14 Management Principles from the World's Greatest Manufacturer. New York McGraw-Hill Education.

Liker JK & Hoseus M (2008) Toyota culture: the heart and soul of the Toyota way. New York, McGraw-Hill.

Liskin O, Schneider K, Fagerholm F, & Münch J (2014) Understanding the role of requirements artifacts in kanban. Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering: 56–63.

Maples C (2009) Enterprise agile transformation: the two-year wall. Proceedings of IEEE Agile Conference: 90–95.

Maassen O, & Sonnevelt J (2010) Kanban at an Insurance Company (Are You Sure?). Proceedings of international Conference on Agile Software Development. Berlin Heidelberg, Springer: 297–306

Magee D (2008) How Toyota became #1: Leadership Lessons from the World's Greatest Car Company New York, Penguin Group.

Mandic V, Oivo M, Rodriguez P, Kuvaja P, Kaikkonen H & Turhan B (2010) What is flowing in Lean Software Development? Proceedings of the 1st International Conference on Lean Enterprise Software and Systems: 72–84.

Mandić V, Markkula J & Oivo M (2009) Towards multi-method research approach in empirical software engineering. Proceedings of Product-Focused Software Process Improvement. Berlin Heidelberg, Springer: 96–110.

Mehta M, Anderson D & Raffo D (2008) Providing value to customers in software development through lean principles. Software Process: Improvement and Practice 13(1): 101–109.

Middleton P (2001) Lean software development: two case studies. Software Quality Journal 9(4): 241–252.

Middleton P & Joyce D (2012) Lean software management: BBC Worldwide case study. IEEE Transactions on Engineering Management 59(1): 20–32.

Middleton P & Sutton J (2005) Lean software strategies: proven techniques for managers and developers. productivity. New York CRC Productivity Press, a division of Kraus Productivity organisation, Ltd.

Mujtaba S, Feldt R & Petersen K (2010) Waste and lead time reduction in a software product customization process with value stream maps. Proceedings of IEEE 21st Australia Software Engineering Conference: 139–148.

Münch J, Armbrust O, Kowalczyk M & Soto M (2012) Software process definition and management. The Fraunhofer IESE Series on Software and Systems Engineering Berlin Heidelberg, Springer-Verlag.

Moe NB (2013) Key Challenges of Improving Agile Teamwork. Proceedings of International Conference on Agile Software Development: 76–90.

Neely S, & Stolt S (2013) Continuous delivery? easy! just change everything (well, maybe it is not that easy). Proceedings of IEEE Agile Conference: 121-128.

N4S (2014) URI: http://www.n4s.fi/en/. Cited 2016/05/27.

Nikitina N, & Kajko-Mattsson M (2011) Developer-driven big-bang process transition from Scrum to Kanban. Proceedings of the International Conference on Software and Systems Process ACM: 159–168.

Norrmalm T (2011) Achieving lean software development: implementation of agile and lean practices in a manufacturing-oriented organization URI http://www.diva-portal.org/smash/get/diva2:400627/FULLTEXT01.pdf Cited 2016/06/28.

Nurdiani I, Börstler J, & Fricker SA (2016). The Impacts of Agile and Lean Practices on Project Constraints: A Tertiary Study. Journal of Systems and Software 119(C): 162–183.

Ohno T (1988) Toyota production system: Beyond Large-Scale Production New York, CRC Press.

Oza N, Fagerholm F, & Münch J (2013) How does Kanban impact communication and collaboration in software engineering teams?. Proceedings of IEEE 6th International Workshop on Cooperative and Human Aspects of Software Engineering: 125–128.

Patton MQ (1990) Qualitative evaluation and research methods. SAGE Publications, Inc.

Pernstål J, Feldt R & Gorschek T (2013) The Lean Gap: a review of Lean approaches to large-scale software systems development. Journal of Systems and Software 86(11): 2797–2821.

Petersen K, & Wohlin C (2009) A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. Journal of systems and software 82(9): 1479–1490.

Petersen K, & Wohlin C (2011) Measuring the flow in lean software development. Journal of Software: Practice and Experience 41(9): 975–996.

Poppendieck M & Cusumano M (2012) Lean software development: a tutorial. IEEE Software 29(5): 26–32.

Poppendieck M & Poppendieck T (2003) Lean software development: an Agile toolkit. Boston MA, Addison Wesley.
Poppendieck M & Poppendieck T (2007) Implementing Lean software development: from concept to cash. Boston MA, Addison-Wesley.
Poppendieck M & Poppendieck T (2009) Leading Lean software development: results are not the point. Boston MA, Pearson Education.
Rahman, N. A. A., Sharif, S. M., & Esa, M. M. (2013). Lean manufacturing case study with Kanban system implementation. International Conference on Economics and Business Research, Elsevier Procedia Economics and Finance 7:174–180.
Radnor ZJ, & Boaden R (2004) Developing an understanding of corporate anorexia. International Journal of Operations & Production Management 24(4): 424–440.
Reinertsen DG (2009) The principles of product development flow: second generation Lean product development. Redondo Beach CA Celeritas Publishing.
Rodríguez P (2013) Combining Lean thinking and Agile software development: how do software-intensive companies use them in practice? PhD thesis URI: http://herkules.oulu.fi/isbn978952620332. Cited 2016/02/18.
Rother M, & Shook J (1999) Learning to see: value stream mapping to add value and eliminate MUDA. Cambridge MA, Lean Enterprise Institute.
Rodríguez P, Mikkonen K, Kuvaja P, Oivo M & Garbajosa J (2013) Building Lean thinking in a telecom software development organization: strengths and challenges. Proceedings of International Conference on Software and System Process: 98–107.
Runeson P & Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering an International Journal 14(2): 131–164.
Rodríguez P, Partanen J, Kuvaja P, & Oivo M (2014) Combining lean thinking and agile methods for software development: A case study of a Finnish provider of wireless embedded systems detailed. IEEE proceedings of 47th Hawaii International Conference on System Sciences: 4770–4779.
Rodríguez P, Mikkonen K, Kuvaja P, Oivo M, & Garbajosa J (2013) Building lean thinking in a telecom software development organization: strengths and challenges. Proceedings of International Conference on Software and System Process: 98–107.
Rodríguez P, Markkula J, Oivo M, & Turula K (2012) Survey on agile and lean usage in Finnish software industry. Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement: 139–148.
Rautiainen K, von Schantz J, Va J (2011) Supporting Scaling Agile with Portfolio Management: Case Paf.com. 44th IEEE Hawaii International Conference on System Sciences: 1–10.
Seaman CB (1999) Qualitative methods in empirical studies of software engineering. Software Engineering, IEEE Transactions on software engineering 25(4): 557–572.
Seikola M, Loisa H & Jagos A (2011) Kanban implementation in a telecom product maintenance. Software Engineering and Advanced Applications. Proceedings of IEEE 37th EUROMICRO Conference: 321–329.
Shalloway A (2010) The real differences between Kanban and Scrum. URI:

http://www.netobjectives.com/blogs/real-differences-between-kanban-and-scrum. Cited 2016/02/10.

Shalloway A, Beaver G & Trott JR (2009) Lean-Agile software development: achieving enterprise agility. Boston MA, Pearson Education, Addison-Wesley.

Shalloway A (2011) Demystifying Kanban. Cutter IT Journal. URI: http://www.netobjectives.com/files/resources/articles/Demystifying-Kanban.pdf Cited 2016/05/08.

Scharlau, B. A. (2013) Games for teaching software development. Proceedings of the 18th ACM conference on Innovation and technology in computer science education: 303–308.

Shinkle CM (2009) Applying the Dreyfus Model of Skill Acquisition to the adoption of Kanban Systems at Software Engineering Professionals (SEP), IEEE proceedings of Agile international conference: 186–191.

Shingo S. (1989) A study of the Toyota Production System: From an Industrial Engineering Viewpoint. New York, Productivity Press.

Shinkle CM & Shihkle CM (2009) Applying the Dreyfus Model of Skill Acquisition to the adoption of Kanban systems at Software Engineering Professionals (SEP). IEEE Agile Conference: 186–191.

Shull F, Singer J & Sjøberg DI (2008) Guide to advanced empirical software engineering. London Springer-Verlag.

Sjøberg DI, Johnsen A & Solberg J (2012) Quantifying the effect of using Kanban versus Scrum: a case study. IEEE Software 29(5): 47–53.

Stone, K. B. (2012). Four decades of lean: a systematic literature review. International Journal of Lean Six Sigma 3(2): 112–132.

Sugimori Y, Kusunoki K, Cho F & Uchikawa S (1977) Toyota production system and Kanban system materialization of just-in-time and respect-for-human system. International Journal of Production Research 15(6): 553–564.

Taylor S & Bogdan R (1984) Introduction to qualitative research methods: the search for meanings, New York Wiley-Interscience publication John Wiley & Sons.

Thun, J. H., Drüke, M., & Grübner, A. (2010) Empowering Kanban through TPS-principles–an empirical analysis of the Toyota Production System. International Journal of Production Research 48(23):7089–7106.

Turner R, Lane J, Ingold D & Madachy R (2013) A Lean approach to improving SE visibility in large operational systems evolution. Monterey CA, Naval Postgraduate School.

Turner R (2014) Value-based scheduling in system of systems evolution. Proceedings of 9th IEEE International Conference on System of Systems Engineering: 301–306.

Tokatli, N. (2008) Global sourcing: insights from the global clothing industry—the case of Zara, a fast fashion retailer. Oxford Journals: Journal of Economic Geography 8(1): 21–38.

Terlecka, K. (2012) Combining Kanban and Scrum - Lessons from a Team of Sysadmins. Proceedings of international agile conference: 99–102.

Tripathi N, Rodríguez P, Ahmad MO, & Oivo M (2015) Scaling Kanban for Software Development in a Multisite Organization: Challenges and Potential Solutions. Proceedings of international conference on agile software development: 178–190.

Venables, M. (2005) Boeing: going for lean [lean manufacturing]. The Institution of Engineering and Technology, IEE Manufacturing Engineer 84(4): 26–31.

Versionone (2016) The 10th Annual State of Agile Survey. Annual State of Agile Survey. URI: http://stateofagile.versionone.com/ Cited 2016/06/05.

Wang X, Conboy K & Cawley O (2012) Leagile software development: an experience report analysis of the application of Lean approaches in Agile software development. Journal of System and Software 85(6): 1287–1299.

Wang X, Lane M & Conboy K (2011) From Agile to Lean: the perspectives of the two Agile online communities of interest. 19th European Conference on Information Systems. paper 209.

West D & Hammond JS (2010) The Forrester wave: Agile development management tools. Q2 2010. Cambridge MA Forrester Research.

Wijewardena T (2011) Do you dare to ask your HR manager to practice Kanban? The experience report of an offshore software company in Sri Lanka introducing Agile practices into its human resource (HR) department. Proceedings of Agile conference: 161–167.

Williams L (2012) What Agile teams think of Agile principles. ACM Magazine of Communications 55(4): 71–76.

Wohlin C & Aurum A (2014) Towards a decision-making structure for selecting a research design in empirical software engineering. Empirical Software Engineering an International Journal: 1–29.

Womack JP & Jones DT (1996) Lean thinking: Banish waste and create wealth in your organisation. New York, Rawson Associates.

Womack JP, Jones DT & Roos D (1990) The Machine That Changed the World: The Story of Lean Production: How Japan's Secret Weapon in the Global Auto Wars Will Revolutionize Western Industry. New York, Rawson Associates.

Wood M, Daly J, Miller J & Roper M (1999) Multi-method research: an empirical investigation of object-oriented technology. Journal of Systems and Software, Elsevier 48(1): 13–26.

Yin RK (2009) Case study research: design and methods. Thousand Oaks California, Sage Publications.

Ziskovsky B, & Ziskovsky J (2007) Doing more with less–Going Lean in education. A White Paper on Process Improvement in Education, Lean Education Enterprises Inc. Shoreview, Minnesota: 1–19.

Zhang Y, Qu T, Ho O, & Huang GQ (2011) Real-time work-in-progress management for smart object-enabled ubiquitous shop-floor environment. International Journal of Computer Integrated Manufacturing 24(5): 431–445.

Zang JJ (2011) A Never Ending Battle for Continuous Improvement. In proceedings of international conference on agile software development, Springer Berlin Heidelberg. 282–289.

Appendix 1 PAPER II PRIMARY STUDIES

Akturk, M., & Erhun, F. (1999). An overview of design and operational issues of kanban systems. International Journal of Production Research, 37(17), 3859-3881.

Ardalan, A. (1997). Analysis of local decision rules in a Dual Kanban flow shop. Decision Sciences, 28(1), 195-211.

Bonvik, A. M., Couch, C., & Gershwin, S. B. (1997). A comparison of production-line control mechanisms. International Journal of Production Research, 35(3), 789-804.

Bitran, G. R., & Chang, L. (1987). A mathematical programming approach to a deterministic kanban system. Management Science, 33(4), 427-441.

Bollon, J., Di Mascolo, M., & Frein, Y. (2004). Unified framework for describing and comparing the dynamics of pull control policies. Annals of Operations Research, 125(1-4), 21-45.

Bonvik, A. M., & Gershwin, S. B. (1996). Beyond kanban: Creating and analyzing lean shop floor control policies. Proceeding of the Manufacturing and Service Operations Management Conference, 46-51.

Baynat, B., Buzacott, J. A., & Dallery, Y. (2002). Multiproduct kanban-like control systems. International Journal of Production Research, 40(16), 4225-4255.

Chang, T., & Yih, Y. (1994). Generic kanban systems for dynamic environments. The International Journal of Production Research, 32(4), 889-902.

Chang, T., & Yih, Y. (1994). Determining the number of kanbans and lotsizes in a generic kanban system: A simulated annealing approach. The International Journal of Production Research, 32(8), 1991-2004.

Dyck, H., Johnson, R. A., & Varzandeh, J. (1988). Transforming a traditional manufacturing system into a just-in-time system with kanban. Proceedings of the 20th Conference on Winter Simulation, 616-623.

Duri, C., Frein, Y., & Di Mascolo, M. (2000). Comparison among three pull control policies: Kanban, base stock, and generalized kanban. Annals of Operations Research, 93(1-4), 41-69.

Dallery, Y., & Liberopoulos, G. (2000). Extended kanban control system: Combining kanban and base stock. IIE Transactions, 32(4), 369-386.

Esparrago Jr, R. A. (1988). Kanban. Production and Inventory Management Journal, 29(1), 6-10.

Fearon, P. A. (1993). Inventory controlled environment (I.C.E.) just-in-time at national semiconductor. Proceedings of the Advanced Semiconductor Manufacturing Conference and Workshop (ASMC), IEEE/SEMI, 34-38.

Farahmand, K., & Heemsbergen, B. L. (1994). Floor inventory tracking of a kanban production system. Proceedings of the Simulation Conference, Winter, 1027-1034.

Frein, Y., Di Mascolo, M., & Dallery, Y. (1995). On the design of generalized kanban control systems. International Journal of Operations & Production Management, 15(9), 158-184.

Framinan, J. M., González, P. L., & Ruiz-Usano, R. (2003). The CONWIP production control system: Review and research issues. Production Planning & Control, 14(3), 255-265.

Gravel, M., & Price, W. L. (1988). Using the kanban in a job shop environment. The International Journal of Production Research, 26(6), 1105-1118.

Golhar, D. Y., & Stamm, C. L. (1991). The just-in-time philosophy: A literature review. The International Journal of Production Research, 29(4), 657-676.

Gupta, Y. P., & Gupta, M. C. (1989). A system dynamics model for a multi-stage multi-line dual-card JIT-kanban system. The International Journal of Production Research, 27(2), 309-352.

Gstettner, S., & KUHN, H. (1996). Analysis of production control systems kanban and CONWIP. International Journal of Production Research, 34(11), 3253-3273.

Geraghty, J., & Heavey, C. (2005). A review and comparison of hybrid and pull-type production control strategies. OR Spectrum, 27(2-3), 435-457.

Gupta, S., & Al-Turki, Y. (1998). The effect of sudden material handling system breakdown on the performance of a JIT system. International Journal of Production Research, 36(7), 1935-1960.

Gupta, S. M., Al-Turki, Y. A., & Perry, R. F. (1999). Flexible kanban system. International Journal of Operations & Production Management, 19(10), 1065-1093.

González-R, P. L., Framinan, J. M., & Pierreval, H. (2012). Token-based pull production control systems: An introductory overview. Journal of Intelligent Manufacturing, 23(1), 5-22.

Huang, C., & Kusiak, A. (1996). Overview of kanban systems. International journal of computer integrated manufacturing, 9(3), 169-189.

Huang, M., Wang, D., & Ip, W. (1998). A simulation and comparative study of the CONWIP, kanban and MRP production control systems in a cold rolling plant. Production Planning & Control, 9(8), 803-812.

Im, J. H., & Schonberger, R. J. (1988). The pull of kanban. Production and Inventory Management Journal, 29(4), 54-58.

Im, J. H. (1989). How does kanban work in American companies. Production and Inventory Management Journal, 30(4), 22-24.

Kimura, O., & Terada, H. (1981). Design and analysis of pull system, a method of multi-stage production control. The International Journal of Production Research, 19(3), 241-253.

Kotani, S. (2007). Optimal method for changing the number of kanbans in the e-kanban system and its applications. International Journal of Production Research, 45(24), 5789-5809.

Kouri, I., Salmimaa, T., & Vilpola, I. (2008). The principles and planning process of an electronic kanban system. Novel algorithms and techniques in telecommunications, automation and industrial electronics, 99-104.

Kumar, C. S., & Panneerselvam, R. (2007). Literature review of JIT-KANBAN system. The International Journal of Advanced Manufacturing Technology, 32(3-4), 393-408.

Kizilkaya, E., & Gupta, S. M. (1998). Material flow control and scheduling in a disassembly environment. Computers & Industrial Engineering, 35(1), 93-96.
Khojasteh-Ghamari, Y. (2009). A performance comparison between kanban and CONWIP controlled assembly systems. Journal of Intelligent Manufacturing, 20(6), 751-760.
LEE, L. (1987). Parametric appraisal of the JIT system. International Journal of Production Research, 25(10), 1415-1429.
Lambrecht, M., & Decaluwe, L. (1988). JIT and constraint theory: The issue of bottleneck management. Production and Inventory Management Third Quarter, 29(3), 61-66.
Lavoie, P., Gharbi, A., & Kenne, J. (2010). A comparative study of pull control mechanisms for unreliable homogenous transfer lines. International Journal of Production Economics, 124(1), 241-251.
Liberopoulos, G., & Dallery, Y. (2000). A unified framework for pull control mechanisms in multi-stage manufacturing systems. Annals of Operations Research, 93(1-4), 325-355.
Marek, R. P., Elkins, D. A., & Smith, D. R. (2001). Manufacturing controls: Understanding the fundamentals of kanban and CONWIP pull systems using simulation. Proceedings of the 33nd Conference on Winter Simulation, 921-929
Miltenburg, J., & Wijngaard, J. (1991). Designing and phasing in just-in-time production systems. The International Journal of Production Research, 29(1), 115-131.
Moeeni, F., Sanchez, S., & Vakha Ria, A. (1997). A robust design methodology for kanban system design. International Journal of Production Research, 35(10), 2821-2838.
Sugimori, Y., Kusunoki, K., Cho, F., & Uchikawa, S. (1977). Toyota production system and kanban system materialization of just-in-time and respect-for-human system. The International Journal of Production Research, 15(6), 553-564.
Spearman, M. L., Woodruff, D. L., & Hopp, W. J. (1990). CONWIP: A pull alternative to kanban. The International Journal of Production Research, 28(5), 879-894.
Savsar, M. (1996). Effects of kanban withdrawal policies and other factors on the performance of JIT systems—a simulation study. International Journal of Production Research, 34(10), 2879-2899.
Sriparavastu, L., & Gupta, T. (1997). An empirical study of just-in-time and total quality management principles implementation in manufacturing firms in the USA. International Journal of Operations & Production Management, 17(12), 1215-1232.
Spencer, M. S., & Larsen, D. (1998). Kanban implementation between a heavy manufacturing department and foundry suppliers. Production Planning & Control, 9(3), 311-316.
Takahashi, K., & Nakamura, N. (1998). Ordering alternatives in JIT production systems. Production Planning & Control, 9(8), 784-794.
Takahashi, K., & Nakamura, N. (2002). Comparing reactive kanban and reactive CONWIP. Production Planning & Control, 13(8), 702-714.
Takahashi, K., & Nakamura, N. (2002). Decentralized reactive kanban system. European Journal of Operational Research, 139(2), 262-276.

Widyadana, G. A., Wee, H., & Chang, J. (2010). Determining the optimal number of kanban in multi-products supply chain system. International Journal of Systems Science, 41(2), 189-201.

Yang, K. K. (2000). Managing a flow line with single-kanban, dual-kanban or conwip. Production and Operations Management, 9(4), 349-366.

Original publications

I Ahmad MO, Markkula J, & Oivo M (2013) Kanban in software development: A systematic literature review. IEEE 39th Euromicro Conference on Software Engineering and Advanced Application: 9–16.

II Ahmad MO, Markkula J, & Oivo M & Adeyemi B (2015) Kanban in Industrial Engineering and Software Engineering: A systematic literature review. 10th International Conference on Software Engineering Advances: 234–241.

III Ahmad MO, Markkula J, Oivo M, & Kuvaja P (2014) Usage of Kanban in Software Companies - An empirical study on motivation, benefits and challenges. 9th International Conference on Software Engineering Advances: 150–155.

IV Ahmad MO, Kuvaja P, Markkula J, & Oivo M (2016). Transition of software maintenance teams from Scrum to Kanban. IEEE 49th Hawaii International Conference on System Sciences: 5427–5436.

V Ahmad MO, Lwakatare LE, Oivo M, Kuvaja P, & Markkula J (2016) Portfolio management and Kanban: An empirical investigation with Agile and Lean software companies. Journal of Software: Evolution and Process (Wiley) (In press)

Original publications are not included in the electronic version of the dissertation.

667. Tolkkinen, Mari (2016) Multi-stressor effects in boreal streams : disentangling the roles of natural and land use disturbance to stream communities

668. Kaakinen, Juhani (2016) Öljyllä ja raskasmetalleilla pilaantuneita maita koskevan ympäristölainsäädännön ja lupamenettelyn edistäminen kemiallisella tutkimuksella

669. Huttunen, Kaisa-Leena (2016) Biodiversity through time : coherence, stability and species turnover in boreal stream communities

670. Rönkä, Nelli (2016) Phylogeography and conservation genetics of waders

671. Fucci, Davide (2016) The role of process conformance and developers' skills in the context of test-driven development

672. Manninen, Outi (2016) The resilience of understorey vegetation and soil to increasing nitrogen and disturbances in boreal forests and the subarctic ecosystem

673. Pentinsaari, Mikko (2016) Utility of DNA barcodes in identification and delimitation of beetle species, with insights into COI protein structure across the animal kingdom

674. Lassila, Toni (2016) In vitro methods in the study of reactive drug metabolites with liquid chromatography / mass spectrometry

675. Koskimäki, Janne (2016) The interaction between the intracellular endophytic bacterium, Methylobacterium extorquens DSM13060, and Scots pine (Pinus sylvestris L.)

676. Ronkainen, Katri (2016) Polyandry, multiple mating and sexual conflict in a water strider, Aquarius paludum

677. Pulkkinen, Elina (2016) Chemical modification of single-walled carbon nanotubes via alkali metal reduction

678. Runtti, Hanna (2016) Utilisation of industrial by-products in water treatment : carbon-and silicate-based materials as adsorbents for metals and sulphate removal

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

681. Kelanti, Markus (2016) Stakeholder analysis in software-intensive systems development

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

UNIVERSITY OF OULU P.O. Box 8000 FI-90014 UNIVERSITY OF OULU FINLAND

ACTA UNIVERSITATIS OULUENSIS

UNIVERSITY of OULU
OULUN YLIOPISTO

ISBN 978-952-62-1407-8 (Paperback)
ISBN 978-952-62-1408-5 (PDF)
ISSN 0355-3191 (Print)
ISSN 1796-220X (Online)