The Journal of Systems and Software 85 (2012) 1287–1299

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

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

"Leagile" software development: An experience report analysis of the application of lean approaches in agile software development

Xiaofeng Wang[a,*], Kieran Conboy[b], Oisin Cawley[c]

[a] Free University of Bozen/Bolzano, Italy
[b] School of Information Systems, Technology and Management, UNSW, Sydney 2052, Australia
[c] Lero, The Irish Software Engineering Research Centre, Ireland

ARTICLE INFO

Article history:
Received 1 September 2011
Received in revised form 27 January 2012
Accepted 31 January 2012
Available online 15 February 2012

Keywords:
Agile software development
Lean software development
Scrum
Leagile
Kanban
Experience report
Software engineering

ABSTRACT

In recent years there has been a noticeable shift in attention from those who use agile software development toward lean software development, often labelled as a shift "from agile to lean". However, the reality may not be as simple or linear as this label implies. To provide a better understanding of lean software development approaches and how they are applied in agile software development, we have examined 30 experience reports published in past agile software conferences in which experiences of applying lean approaches in agile software development were reported. The analysis identified six types of lean application. The results of our study show that lean can be applied in agile processes in different manners for different purposes. Lean concepts, principles and practices are most often used for continuous agile process improvement, with the most recent introduction being the kanban approach, introducing a continuous, flow-based substitute to time-boxed agile processes.

1. Introduction

Agile methods have become highly prevalent since the Agile Manifesto[1] emerged in early 2001 as a response to the inefficiency of existing software development methods in rapidly changing environments (Highsmith, 2002). In agile literature, *agile methods* generally denote a family of methods under the umbrella of the Agile Alliance, including: eXtreme Programming (XP, Beck, 1999), Scrum (Schwaber and Beedle, 2002), Dynamic Systems Development Method (DSDM, http://www.dsdm.org), Crystal Methods (Cockburn, 2001), Feature-Driven Development (FDD, Coad and Palmer, 2002), Lean Development (Charette, 2003) and Adaptive Software Development (ASD, Highsmith, 2002). Although differing in specific techniques, these methods have much in common, including short iterative life cycles, quick and frequent feedback from customers, and constant learning. Among them, Scrum and XP/Scrum hybrid are by far the most widely adopted in the past decade (VersionOne, 2010).

In recent years however the agile community has started to look toward lean software development approaches, in addition to agile methods such as XP and Scrum. Emerging lean conferences (e.g. Lean Software and Systems Consortium conference series, Lean Enterprise Software and Systems conference) show that lean adoption is spreading. Lean approaches are claimed to be "the next wave of software process".[2] Even though originally lean software development was viewed as just another agile method (Highsmith, 2002; Dybå and Dingsøyr, 2009), there is an increasing focus on lean and it is viewed as being a method category in itself rather than an instance of agile methods (Hibbs et al., 2009). More and more people advocate "from agile to lean" (Hiranabe, 2008), and the application of lean approaches in agile software development. Some claim that lean software development provides the theory behind agile practices (Poppendieck and Poppendieck, 2003, 2006). Others argue that lean is a necessary progression for organisations planning to scale up agility from the project or team level to the organisational level, which agile methods fail to address satisfyingly (Smits, 2007). "Scrumban" is a term coined for a software production model based on Scrum and kanban (Ladas, 2009). It is believed to be especially suited for maintenance projects or projects with frequent and unexpected user stories or programming errors.

* Corresponding author at: Dominikanerplatz 3 piazza Domenicani, I-39100 Bozen/Bolzano, Italy. Tel.: +39 0471 016 181.
E-mail addresses: xiaofeng.wang@unibz.it, xiaofeng.wang@ul.ie (X. Wang), k.conboy@unsw.edu.au (K. Conboy), oisin.cawley@lero.ie (O. Cawley).

[1] http://www.agilealliance.org.

[2] http://atlanta2010.leanssc.org/.

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2012.01.061

In such cases the time-boxed sprints of the Scrum model are of no appreciable use, but Scrum's daily meetings and other practices can be applied, depending on the team and the situation at hand. Using these methods, the team's workflow is directed in a way that allows for minimum earliest completion time for each user story or programming error, but on the other hand ensures each team member is constantly employed (Ladas, 2009).

Despite the shifting focus of practice from agile software development represented by Scrum and XP to lean software development, few studies have been conducted that can offer a good understanding of the application of lean approaches in agile software development, due to the fact that lean in software development is a nascent research area yet to be explored. Naylor et al. (1999) coin the term "leagility" whereby principles were established for combining both lean and agile in a supply chain strategy. We borrow the term and use "leagile" software development to denote the software development processes that include both agile and lean elements.

The purposes of our study, consequently, are (1) *to provide a better understanding of the application strategies of lean approaches in agile software development*, and (2) *to demonstrate how these strategies are being implemented in practice*. To achieve these goals, we have conducted a secondary data analysis of a set of experience reports in which the real world application of lean approaches in agile software development is evidenced. These experience reports were published in the past Agile and XP conference series. Patterns of lean application in agile software development that recurred in these experience reports were identified. The results are intended to be useful for both research and practice.

The remaining part of the paper is organised as follows. Section 2 introduces basic lean concepts and lean principles and practices for software development. This is followed by a section that reviews the relevant literature on the application of lean approaches in agile software development. Section 4 describes the research approach employed. Then the findings are reported in the next section, and further reflected upon in the discussion section. The paper ends with concluding remarks.

2. Lean software development

It is well acknowledged that the terms 'agile' and 'lean' are poorly defined in the software development literature, and the use of these terms is often poorly considered, multi-dimensional, ambiguous and inconsistent (Conboy, 2009). For the purpose of our study, however, we need to make it as explicit as possible what we mean by lean software development. Corresponding to relatively well-accepted agile values, principles and practices, we consider that lean software development also involve three main elements: lean concepts, lean principles and lean practices.

2.1. Lean concepts

Leanness, like agility, is not a term unique to software development. It has a much older origin rooted in other disciplines, with most literature tracing the origins back to the Toyota Production System (TPS) in the 1950s (Ohno, 1988). However, it did not make a significant impact in the mainstream literature until MIT's (Massachusetts Institute of Technology) 5-year study of the automotive industry identified lean as a source of huge productivity differences between the US and Japan (Womack et al., 1990).

Lean thinking is a way of thinking that enables organisations to "specify value, line up value-creating actions in the best sequence, conduct these activities without interruption whenever someone requests them, and perform them more and more effectively" (Womack and Jones, 1996). Five inherently interlinked guiding lean concepts underpin lean thinking:

- *Value*: It is defined by the customer and it is paramount to have a clear understanding of what that is.
- *Value stream*: A map that identifies every step in the process and categorises each step in terms of the value it adds.
- *Flow*: It is important that the production process flows continuously.
- *Pull*: Customer orders pull product, ensuring nothing is built before it is needed.
- *Perfection*: Striving for perfection in the process by continuously identifying and removing waste.

Table 1
Lean principles relevant to software development.

Lean Software Development Principles (Poppendieck and Poppendieck, 2003)[a]	The Principles of Product Development Flow (Reinertsen, 2009)	The Kanban Principles (Anderson, 2010)
• Eliminate waste • Build quality in • Create knowledge • Defer commitment • Deliver fast • Respect people • Optimise the whole	• Use an economic view • Manage queues • Exploit variability • Reduce batch size • Apply WIP (Work in Progress) constraints • Control flow under uncertainty • Use fast feedback • Decentralise control	• Visualise the workflow • Limit WIP • Manage flow • Make process policies explicit • Improve collaboratively (using models and the scientific method)

[a] These seven principles have been rephrased and rearranged by The Poppendiecks. The newest version can be seen at http://www.poppendieck.com/.

2.2. Lean principles

The primary focus and guiding principle of lean is the identification and elimination of waste from the process with respect to customer value. In Japanese this waste is referred to as *muda*, although other terms such as *mura* (unevenness), and *muri* (overburden) are also used. Lean thinking classifies work into three categories: value-adding activities, required non-value adding activities, and non-value adding activities. By mapping out the process using a value stream map, those process steps which do not contribute to creating value can be identified and eliminated. It is worth noting that the concept of waste can be quite broad and context dependent. In the domain of software development, the types of waste can be interpreted as: *extra features*, *waiting*, *task switching*, *extra processes*, *partially done work*, *movement*, *defects* and *unused employee creativity* (Poppendieck and Poppendieck, 2003; Hibbs et al., 2009).

The contemporary understanding of lean software development is largely driven by practitioners writings (e.g., Poppendieck and Poppendieck, 2003, 2006; Hibbs et al., 2009; Reinertsen, 2009; Anderson, 2010). Maintaining the core intent of lean, different lean principles for software development have been proposed and are constantly evolving. Table 1 lists several sets of lean principles better known in agile community. These sets of lean principles overlap to large extent which reflects the core and essence of lean approaches.

It is worth noting that the kanban approach is the most recent addition to the agile and lean software development affray. Again, it gets its name from the world of lean manufacturing. A kanban system is "*a production control system for just-in-time production and making full use of workers' capabilities*" (Sugimori et al., 1977). The core objective of the kanban system is to minimise the amount

of WIP. Excess WIP is one form of waste from a lean perspective. Work should be "pulled" through the system as it is needed, as opposed to "pushing" it through. Only when a downstream process is ready and needs to do some more work does it pull work from an upstream process. The signalling between upstream and downstream processes is typically done via some sort of coloured card which physically travels between processes. The aim is to keep the process flowing at an even but continuous rate. This is achieved by controlling the number of kanban cards which are in circulation within the process.

2.3. Lean practices

Some lean software development practices have already been well established in agile methods, and regarded as agile practices as well, even though they have origins that can be traced back to pre-agile days. For example, back in the 1970s, Michael Fagan, while working for IBM, formalised the practice of code reviews in a technique known as Fagan's Inspections: "*Because errors were identified and corrected in groups [early in the process] rather than found one-by-one during subsequent work and handled at the higher cost incumbent in later rework, the overall amount of error rework was minimised, even within the coding operation*" (Fagan, 1976, p. 187). Code inspections support one of the cornerstones of what lean software development is founded on, finding and fixing defects early in the development process.

In order to help categorise the empirical data in terms of lean approaches, we generated a list of what we considered to be lean specific practices relevant to software development, as shown in Table 2. Due to the overlapping nature of lean and agile practices, we limited the practices to those we found were less represented in the agile literature but were recurring themes within a lean context. This categorisation was somewhat subjective. However, the final list was agreed by consensus among the researchers and offers a good starting point for cataloguing lean practices.

Table 2
Lean practices relevant to software development.

Lean software development practices
- Address bottlenecks (Liker, 2003; Goldratt, 1992, 1997; Middleton et al., 2005; Poppendieck and Poppendieck, 2003)
  - Cumulative Flow Diagram (CFD)
- Avoid too much local optimisation (Poppendieck and Poppendieck, 2003)
- Defer decision making (Thimbleby, 1988; Poppendieck and Poppendieck, 2003)
- Develop appropriate incentives/rewards (Ambler and Kroll, 2007)
- Hansei: relentless self-reflection, to acknowledge one's own mistakes and to commit to making improvements (Liker and Hoseus, 2008).
- Heijunka: workload levelling, production smoothing. It aims at reducing muda (Liker, 2003; Middleton et al., 2005).
- Hide individual performance (Poppendieck and Poppendieck, 2003)
- Jidoka: intelligent automation, automation with a human touch. People should not serve machines but vice versa (Liker, 2003; Liker and Hoseus, 2008).
- Kaikaku: radical improvement within a limited time (Womack and Jones, 1996).
- Kaizen: continuous improvement to establish a smoother flow (Liker, 2003; Hibbs et al., 2009; Joyce and Schechter, 2004).
- Kano analysis: link voice of the customer to requirements (Middleton et al., 2005; Raffo et al., 2010).
- Make everything transparent (Womack and Jones, 1996):
  - Make project status highly visible
  - Visualise all work items
- Measure and manage (Anderson and Garber, 2007)
  - Employ queuing theory (Reinertsen, 1997; Goldratt, 1997, 1992) but measure the right things (Reinertsen, 1997)
  - First-In-First-Out (FIFO) queue
- Move variability downstream (Poppendieck and Poppendieck, 2003)
- Plan-Do-Check-Act (PDCA) cycle (Deming, 1986)
- Poka-Yoke: defect detection and prevention (Robinson, 1997).
- Pragmatic governance (enable first, manage/control second) (Ambler and Kroll, 2007)
- Pull the andon cord: promote a "safe to failure" environment and instil a "stop the line" mentality (Poppendieck and Poppendieck, 2003; Womack et al., 1990)
- Quality Function Deployment: transform the voice of the customer into engineering characteristics and appropriate test methods (Raffo et al., 2010).
- Reduce slack (Middleton, 2001)
- Root cause analysis
  - The 5 whys? (Womack et al., 1990)
- Use pull systems
  - Kanban board, Limitied WIP, CONWIP (Sugimori et al., 1977; Bradley, 2007; Kniberg and Skarin, 2010)
  - Batch control processing (Bradley, 2007), Minimal Marketable Feature (MMF)
- Value stream mapping: analyse and design the workflow required to bring a software or service to a customer (Womack and Jones, 1996; Liker, 2003; Poppendieck and Poppendieck, 2003; Mujtaba et al., 2010).

3. Application of lean approaches in agile software development

Agile and lean are seen as just two different names for the same thing in some software literature. In the study reported in Jalali and Wohlin (2010), for example, no meaningful distinction is made between the two. This study conducted a literature review of agile practices used in global software engineering. The fact that the search string "agile and lean" was used to denote agile practices indicates a lack of distinction between the two. Barton (2009) claims that many organisations that have modified their software development system based on Scrum consider their work to be a lean implementation. Barton argues that even in its simplest state, Scrum uses a lean 'pull' technique to smooth the flow through the system and prevent overloading. Scrum also implements a process for eliminating muda, or waste. If no distinction is made between agile and lean, the application of lean in agile context, if happens, is generally a non-purposeful act of the adopting organisation.

Most literature, however, does consider the differences between agile and lean approaches, thus motivating an analysis of their relationship and a study that analyses the purposeful application of lean approaches in agile software development. Agile methods are believed to be tactical in nature, and therefore the major changes required to become agile must be initiated from the top of the organisation. Organisational strategy becomes the context within which agile processes can operate effectively. Without this strategic piece, agile development is "shunted aside by the organisational forces that seek equilibrium" (Highsmith, 2002). Dall'Agnol et al. (2003) also suggest that agile and lean address a different audience. They believe that XP describes a set of practices primarily designed for use by developers. It is aiming to ease the tension often exhibited between developer and customer due to conflicting aims. Instead, lean management is applied from an upper management perspective, with the objective being the optimisation of activity across the whole entire organisation. Therefore lean management is a top-down approach. Smits (2007) claims that "experience gathered during large scale implementation of agile concepts in software development projects teaches us that the currently popular agile software development methods (like Scrum) do not scale to programme, product and organisation level without change. The fundamentals for changes to these methods are found in lean principles, or: "the future of agile methods is found in its origins." This claim is echoed by industry practitioners (reported in Serignese (2011)) who believe that "lean is both the precursor and future of agile". Similarly, Hibbs et al. (2009) view agile methods as mostly concerned with the specific practice of developing software and the project management that surrounds that software development. They do not generally concern themselves with the surrounding business context in which the software development

Table 3
The application of lean approaches in agile software development.

Types of lean application in agile software development
Non-purposeful combination of agile and lean in software development processes
Purposeful application of lean approaches in agile software development
Lean approaches are applied to the business areas related to software development
Lean approaches are applied to software development processes directly

is taking place. Instead, lean principles can be applied to any scope, from the specific practice of developing software to the entire enterprise where software development is just one small part.

Poppendieck and Poppendieck (2003, 2006) focus more on the application of lean in software development activities. They claim that lean thinking is principles that guide ideas and insights about software development discipline. Principles are viewed as underlying truths that do not change over time or space, while practices are the application of principles to a particular situation and should differ from one environment to the next and change as a situation evolves. They believe that lean development further expands the theoretical foundations of agile software development by applying well-known and accepted lean principles to software development. Consequently they suggest use lean thinking as guiding principles to develop and adapt agile practices. Morien (2005) also sees that agile project management has roots in the lean thinking which provides strength and credibility to the concept and practice of agile project management. Along the same line, the study of Perera and Fernando (2007) attempts to identify possible improvable parts in agile software development processes, and explores how lean practices could be used to improve them. A hypothesis "agile software process's development can be improved using lean practice techniques" is proposed in their study. They conducted a controlled experiment with university student projects to test the hypothesis and obtained positive evidences to support it. They conclude that applying lean techniques help stabilise the agile development phase especially in later stages of the phase. Instead, Ambler (2009) proposes a governance framework built upon the lean principles that is claimed to enable agility at scale. Lean governance practices such as aligning the team structure with the architecture, risk-based milestones, and staged programme delivery address complexities inherent in large or distributed teams. Other practices such as continuous project monitoring, integrated lifecycle environment, and embedded compliance help to address the additional complexity of regulated environments.

Table 3 is a summary of the possible applications of lean approaches in agile software development suggested in the reviewed literature. The limited yet increasing literature on lean software development depicts a fragmented picture of the strategies of lean application in agile software development. The empirical evidences of such applications and their implementation are yet to be collected in a systematic manner. Our study is one of the earliest attempts to address this knowledge gap.

4. Research approach

To explore *how lean approaches have been applied in agile software development*, we have conducted a secondary data analysis of the real world cases that contain the evidences of lean application in agile software development. Secondary data analysis is the analysis of data that was either collected by individuals other than the researchers that conduct the study, or for some other purpose than the one currently being considered, or often a combination of the two. The sources of secondary data include newspapers, census data, maps, etc. The advantage of using secondary data is that the data collection process can be unobtrusive, fast and inexpensive, even though it needs to be cautioned that there are issues related to data quality control, accuracy of data, etc., which need the attention of the researchers. Secondary data analysis is frequently used in social science research. If it is undertaken with care and diligence, it can provide a cost-effective way of gaining a broad understanding of research questions. It is also often considered a starting point for other research methods, helpful in designing subsequent primary research and can provide a baseline with which to compare the primary data analysis results (Boslaugh, 2007). Given the exploratory nature and early stage of our study, we consider secondary data analysis a feasible way to build initial understanding of the phenomenon under the study.

Table 4
The source conferences and the numbers of the selected experience reports.

Conference	Number of experience reports selected
Agile 2004	2
Agile 2006	4
Agile 2007	3
Agile 2008	5
Agile 2009	4
XP 2010	3
Agile 2011	8
XP 2011	1
Total	30

Agile development has been the subject of several conferences, and some of these conferences have published experience reports which share industry experiences of agile software development. To obtain the secondary data needed for this study, we collected the experience reports that have been published in the agile related conferences from 2000 to 2011 (including the XP Conference series, Agile Conference series and XP/Agile Universe series) and that are publicly available online (Agile 2010 experience reports and the proceedings of XP 2000, XP 2001, XP 2002 and XP Universe 2001 are not publicly available online). For each experience report, we conducted a full-text search for any lean concept, principle or practice as defined in Section 2. If an experience report contained one or more lean keywords, we then read through the report to decide: (i) if it has an agile software development context, and (ii) if it contains explicit evidences of applying one or more lean concepts, lean principles and/or lean practices in the agile context. In total 30 experience reports satisfy the selection criteria and have been included in the secondary data analysis (see Appendix for a list of these experience reports). Table 4 shows the source conferences and number of the selected experience reports per conference.

The 30 selected experience reports focus on different aspects of software development. The angles and detail levels of the description of lean application vary from one experience report to another. This diversity posed a challenge to qualitatively analyse these 30 experience reports at the same depth and in a unified manner. We started with the initial classification scheme of lean application in agile software development presented in Table 3. Combined with our understanding of lean approaches, it acted as a sense-making and categorisation device for the identification of patterns of lean application in agile software development. Meanwhile, we used an open coding process and allowed more detailed patterns emerge from the collected data which enriched and extended the initial classification scheme. Some initial insights gained through analysing a subset of the reports has been reported in Wang (2011). The following section describes the findings of the analysis of all 30 experience reports.

Table 5
The six categories of lean application in agile software development.

Type of lean application	Type code
Non-purposeful combination of agile and lean in software development processes	A
Purposeful application of lean approaches in agile software development	
Lean approaches are applied to the business areas related to software development	
Agile within, lean out-reach: using lean approaches to interact with neighbouring business units while keeping agile development processes internally	B
Lean approaches are applied to software development processes directly	
Lean facilitating agile adoption: before or during the transition process	C
Lean within agile: using various lean elements to improve agile processes	D
From agile to lean: comprehensive application of lean approaches to transform agile processes	E
Synchronising agile and lean: agile team and lean team work in parallel in a synchronised manner	F

5. Findings

The 28 organisations reported in the 30 experience reports (3 experience reports regard the same company) range from small and medium to large and multi-national. They operate in diverse business domains, such as software engineering, telecommunication, finance, healthcare and public administration. Lean approaches have been applied to various agile projects of these organisations, in various manners and to various degrees. In some companies, lean application is limited to a single, or several agile teams or projects. In others instead, it is a company-wide endeavour. Table 5 shows the six categories of lean application in agile software development identified based on the analysis of the 30 experience reports. The distribution of the experience reports per application type is shown in Fig. 1.

A more detailed distribution of the experience reports per type per year is shown in Table 6.

It can be seen that Type D, using lean to improve agile processes, is the most common and frequently appearing lean application type over the past years, with almost a half of the selected experience reports containing the relevant evidences. Another observation is that, Type E seems to be an emerging trend of lean application in the last several years. In the following sub-sections we present in more depth what the six categories of lean application mean and how various lean concepts, principles and practices have been applied in the cases reported in the experience reports.

5.1. Non-purposeful combination of agile and lean

The reporter of [ER13] presented the agile practices that he practiced in a project where he worked as a technical lead (The company's name was not revealed in the report). The practices in his armoury included both agile practices (Keep/Problem/Try Retrospective, Estimate Retrospectives, Iteration Planning, etc.) and lean practice (Task Kanban). Evidently he was practicing a combined approach of agile and lean practices, even though he regarded them as agile practices in general. In this report lean and agile practices were presented in parallel and no specific distinction between the two was made. Therefore it is a case of non-purposeful combination of agile and lean practices. It is interesting to mention that the Japanese background of the development team might be an explanation of the natural and implicit adoption of lean practices.

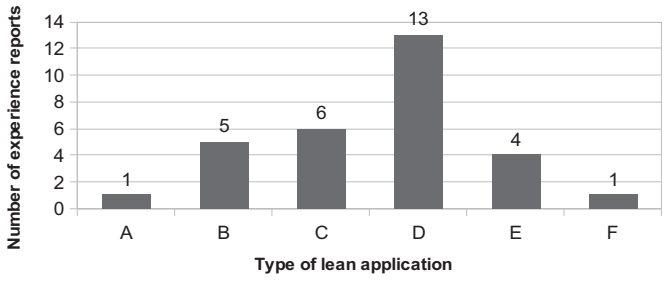

Fig. 1. The distribution of the selected experience reports per lean application type.

5.2. Agile within, lean out-reach

Five experience reports are grouped under this category. Table 7 lists the cases contained in these reports.

In the Cardiac Rhythm Disease Management of Medtronic Inc., when the software group adopted agile development, they quickly realised that not only were they learning something new, but they also needed to learn how to communicate it with other business units ([ER5]). Lean concepts and principles became a convenient choice as a communication tool with the product development organisation since the product development organisation had already implemented a lean initiative. Therefore when communicating with product development organisation, the software group emphasised the principle of eliminating waste, meanwhile implemented customised agile practices like the Customer Role, Stories, Sprint Planning and Release Planning to put focus on value-added activities.

The experience of Ericsson R&D in the Netherlands suggests that agile software development should be implemented as a broader "lean" initiative ([ER12]), which can create involvement of neighbouring units, for example, service and delivery units, product management, market units and customers. This foundation would be an incentive for neighbouring units to cooperate and optimise as a whole, and the resistance to collaborate with agile development would be reduced effectively. Their experience also suggests that lean can help align management quickly:

> **"***In a bottom-up approach line and project management have limited involvement. However, line and project managers are a key in making the change stick, helping people resolve impediments and conflicts, building a learning organization and, above all, showing what is meant with Agile and lean ways of working. Speaking the same language and agreeing on principles like 'build quality in' and many more is highly needed for successful cross unit cooperation.*" ([ER12], p. 158)

An interesting case was reported in [ER2] where a company applied lean thinking and systems thinking on the customer business process in order to understand what features the developers should develop that really delivered business values. Cyrus Innovation is a consulting group in New York City specializing in agile software development, usability design and operational consulting. One of their customers – a restaurant chain – needed to improve aspects of their operations that were breaking due to their rapid growth. Their main objective was reducing operating expenses by cutting down on the food product wasted each day. Cyrus Innovation used XP for development and strongly believed in the power of agile development; however they saw that agile alone was insufficient to make every software project successful:

> "*XP customer will tend to drive development without regard to how features impact the business system from a throughput perspective... If business managers do not adopt analytic techniques that are more synergistic with XP, the developers themselves simply become a local optimization of the software development process. As a result, XP by itself cannot drive overall system improvement and thus by itself cannot make a company sustainable.*" ([ER2], p. 81)

Table 6
The distribution of the selected experience reports per application type per year.

	A	B	C	D	E	F
2004		[ER2]		[ER1]		
2005						
2006		[ER5]	[ER3] [ER4]	[ER6]		
2007			[ER7][ER9]	[ER8]		
2008	[ER13]	[ER12] [ER14]		[ER10] [ER11]		
2009			[ER17]	[ER15] [ER16]	[ER18]	
2010				[ER19]	[ER20][ER21]	
2011		[ER24]	[ER28]	[ER22][ER23][ER25][ER26][ER27]	[ER29]	[ER30]

Projects must be coupled with a complimentary approach to strategy in order to achieve the overall business goals. Rather than accepting whatever the customer demanded to deliver, the project team used lean thinking and techniques, including eliminating waste, value stream mapping, flow, and theory of constraints, to better understand what really delivered business value to the customer. Then the development team used agile development to build quality software that effectively contributed to overall business success.

In DTE Energy, a Fortune 300 company, agile software development and project management encountered the legacy mindset and culture of portfolio management ([ER14]), which meant that the annual budgeting cycle drove a mindset that scope, budget, and schedule must be established up front, often many months before project work began. Success meant delivering on that scope within the budget and schedule commitments. As the IT teams successfully applied agile methods at the project level, they began to address their approach for managing portfolios of projects to increase the amount of value they delivered with their business partners. Lean training was organised for the leadership team, and potential applicable lean techniques identified. They also introduced lean terminology and concepts to help better understand the constraints and how they could reorganise the way they prioritised their commitments and funded their work. As the way forward, one overarching strategy in their IT was to leverage an existing corporate system which is a combination of lean and Six Sigma thinking, seeing, and doing tools and techniques based upon the Plan-Do-Check-Act (PDCA) cycle.

Similarly Exilesoft, an offshore software development company in Sri Lanka which caters to the Scandinavian and Australian markets, applied lean to its Human Resource (HR) department ([ER24]). With the rapid change from a traditional project culture to an agile one and the speedy growth of the project organisation, supporting functionalities such as HR and operations were stressed out and a high degree of negative stress and frustration was created throughout the company:

> "*. . . software companies adapt agile concepts for their development teams rapidly. However, the lack of understanding of such concepts by other facilitating entities of the organization, such as HR department, may create complexities and slow down the expected return of such agile transformation by its production staff.*" ([ER24], p. 166)

Based on the believe that employing the same agile concepts within the HR department would also deliver positive results and that having one work culture across the organisation would remove many challenges faced by the company, the senior management group launched an initiative to introduce agile concepts to its HR department. However, due to the nature of the work in HR, feasibility of implementing Scrum was rejected, including the concepts of time boxing, scheduled releases, story point estimation, sprint burn down, etc. Instead, kanban was seen a better fit for the HR department. Kanban supported everyday planning required for HR functions. It did not demand story point estimation/relative estimation, and allowed to monitor cycle time and set limit to work-in-process, which solved one of the biggest issues in the HR department, preventing it from committing to work on various tasks which would exceed their capacity.

In brief, the five cases show that the agile teams have used purposefully lean approaches to involve and interact with their surrounding environments while keeping the agile processes internally. They have demonstrated lean approaches, especially the lean concepts and principles, could help agile teams to better collaborate with different stakeholders of software development, including neighbouring business units and customers, and extending agile mindsets and practices to non development, organisational level activities.

In the following sub-sections we show how lean approaches have been used purposefully in software development activities directly and demonstrated their effectiveness as software development approaches.

5.3. Lean facilitating agile adoption

Six experience reports contain evidences that lean concepts and principles have been utilised to facilitate agile adoption, either

Table 7
The cases in the "agile within, lean out-reach" category.

Experience report no.	Organisation name	Business domain	Lean elements applied
[ER5]	Medtronic Inc.	Medical device	*Concept*: Value *Principle*: Eliminate waste
[ER12]	Ericsson	Telecommunication	*Principle*: Build quality in
[ER2]	Cyrus Innovation	Agile software development consulting group	*Concept*: Flow *Principle*: Eliminate waste *Practice*: Avoid too much local optimisation, Value stream mapping
[ER14]	DTE Energy	Gas and electric utility services	Lean concepts[a] *Practice*: PDCA cycle
[ER24]	Exilesoft	Software development	*Principle*: Limit WIP *Practice*: Kanban board

[a] Which specific lean concepts were applied is not explained in the report.

Table 8
The cases in the "lean facilitating agile adoption" category.

Experience report no.	Organisation name	Business domain	Lean elements applied
[ER3]	Capital One	Financial service	*Principle*: Eliminate waste *Practice*: Value stream mapping, Root-cause analysis
[ER9]	Systematic	Software systems	*Principle*: Build quality in, Create knowledge, Deliver fast
[ER4]	(*not revealed*)	Automotive engineering consultancy and software	*Concept*: *Self funding transformation*[a], *cost accounting*[a]
[ER7]	Salesforce.com	Enterprise software	*Principle*: Respect people, Deliver fast
[ER17]	SEP	Software solution for regulated industry	*Practice*: Kanban board
[ER28]	Cisco Voice Technology Group	Voice technology and service	*Concept*: Value, Flow *Principle*: *Principles of product flow*[b] *Practice*: Kanban board

[a] Considered lean elements in the report but not included in our definition.
[b] Which specific principles were applied is not explained in the report.

before or during agile transition initiatives. Table 8 shows the profiles of the six cases.

Capital One, a large Fortune 500 financial services company used the lens of lean to evaluate the current delivery process and streamline the business values prior to agile transition ([ER3]). Lean principle, eliminate waste, and practices including value stream mapping, root-cause analysis (5-whys) were applied to analyse the processes of a product or service type before Scrum was chosen as the adopted agile method. According to the experience of the company, lean principles and practices could help in achieving a smoother transition and would increase the likelihood of a successful agile pilot with tangible business metrics.

The experience of Capital One is echoed by that of Systematic, an independent software systems company ([ER9]). Systematic made a strategic decision to use lean as the dominant paradigm for further improvements after achieving CMMI (Capability Maturity Model Integration) Level 5. The company identified Lean Software Development of Poppendieck and Poppendieck (2003) as the lean dialect most relevant to Systematic. The analysis of systematic improvement opportunities and lean causal dependencies led to the decision to seek improvements based on the lean principles of build integrity in, amplify learning and deliver fast. These lean thinking tools gave inspiration to consider Scrum and early testing.

[ER4] describes an interesting experience of the bottom-up, self-funded agile adoption in an engineering consultancy and software company that is involved in the automotive industry "*where analogous concepts under the umbrella term 'lean' are part of the landscape*" and some of the companies they do business with were the originators of lean. Therefore it is natural that the driving force behind a lot of what they did came from an awareness of lean. According to the report, a self funding transformation is one of the key concepts they "stole" from lean. Another concept that they borrowed from lean is that of cost accounting, viewing your software team as a fixed cost overhead. In addition, what inspired them to bootstrap agile practices themselves include also the "inspect and adapt" cycle. Therefore, the success of this self-funded agile adoption can be attributed to the guidance of both lean and agile concepts.

In contrast to the bottom-up, self-funded agile adoption described in [ER4], Salesforce.com took a completely different approach ([ER7]). Salesforce.com has completed an agile transformation of a two hundred person team within a three-month timeframe. During this large and fast "big-bang" agile rollout, lean principles, such as empowered teams and delivering customer value early, were used as key communication tools to communicate the value of changing current behaviour. If the teams were feeling that something was not working the way it should be, they could refer back to the values and reject anything they thought did not correlate with their core values. Lean principles help agile behaviours to stick.

This claim can be supported by the experience of SEP ([ER17]). SEP is a privately held software engineering company with more than 70 employees. It offers full lifecycle software solutions to clients in the medical, aerospace, healthcare and national defence markets. In the process of adopting one of the agile methods, Feature Driven Development, SEP found that it failed to have the desired lasting impact across the entire organisation. However, things changed when a kanban system was implemented later on alongside the agile practices. Kanban provided a more effective vehicle to introduce agile practices and principles in the company. The culture on project teams began to change as they learned the system. And more importantly, the attitudes of team members changed. The implementation of kanban helped the agile mindsets to stick in the company.

The agile transition in Cisco Voice Technology Group (VTG) was also a top-down organisational wide initiative ([ER28]). VTG is a global organisation with three business units, with a total of about 2500 people within the larger Cisco Systems, Inc. When Scrum was to be implemented in the organisation, it was realised that, even though "*... it is often suggested to implement Scrum by using Scrum: create a backlog of process changes, prioritise, and start implementing*", the kanban approach was found more appropriate to implement Scrum in VTG:

> "*we had a vision and model we wanted to implement, we had a backlog of steps to take, and when problems occurred, we would prioritize the issue, put it on the backlog, and address it when it became the highest priority. Sometimes that meant addressing an issue immediately*." ([ER28], p. 274)

An important lesson learned by VTG was that implementing agile methods into an organisation was "*interrupt-driven, not plan driven. We had some interesting hurdles to clear, but once taken they became a strong driver in the change process.*" Other lean concepts, such as value and flow, and the principles of product development flow, also inspired and helped the different organisational units, including the executives, to comprehend the nature of the agile transition happening in the organisation.

As shown in these six cases, regardless the agile adoption style (top-down organisational undertaking or grass-root, bottom-up initiative) or applied before or during agile adoption, lean concepts, principles and practices can smooth the agile transformation process and help agile mindsets to be institutionalised in adopting organisations.

5.4. Lean within agile

Using various lean elements to improve agile processes is a pattern that appears repeatedly in 13 out of 30 experience reports analysed. Table 9 is a list of the 13 cases.

Table 9
The cases in the "lean within agile" category.

Experience report no.	Organisation name	Business domain	Lean elements applied
[ER1]	Government of a major California county	Public administration	*Principle*: Reduce batch size
[ER16]	Canonical	Open source solution, collaborative software systems	*Concept*: Value stream *Practice*: Address bottlenecks, Kaizen
[ER6]	Wireless Data Services Global	Services to mobile companies	*Principle*: Eliminate waste
[ER8]	Sabre Airline Solutions	Product development for airline industry	*Principle*: Eliminate waste *Practice*: Value stream mapping
[ER10]	British Telecom	Telecommunication	*Principle*: Eliminate waste
[ER11]	(*not revealed*)	(*not revealed*)	*Principle*: Eliminate waste
[ER19]	ASR Insurance	Insurance	*Principle*: Visualise the workflow, Limit WIP, Manage flow *Practice*: Kanban board
[ER23]	Fundamo	Mobile financial services products	*Practice*: Kanban board, CFD
[ER27]	SumTotal Des Moines	Learning management system	*Practice*: Value stream mapping, FIFO queue
[ER26]	(*not revealed*)	Financial service	*Practice*: Visualise all work items, Root cause analysis, Kaizen
[ER22]	(*not revealed*)	Telecommunication	*Practice*: Use pull systems, Kanban board, Value stream mapping, Kaizen, Pull the andon cord
[ER15, ER25]	Systematic	Software systems	*Practice*: Kaizen, PDCA cycle, Root cause analysis, Jidoka

Lean concepts and principles have been used as thinking tools to make sense and guide the use and adaptation of agile practices in [ER1]. The Government Workflow Project, a project initiated by the government of a major California county to automate the workflow of key business processes in the criminal justice system, has adopted agile practices incrementally. The project team ended up performing more up-front analysis and using small batch size for estimations as a response to frustrating velocity fluctuations and inconsistent completion of features. Initially the team had doubts that performing additional up-front analysis was against agile principles. However, "*in retrospect, the team realised this practice implements the 'smaller batch size' principle of Lean Software Development, and in fact increased their agility*". Notice that lean was used as a sense making tool retrospectively after a practice was adapted. More lean principles have been applied in a distributed team of 35 developers spanning 5 continents in Canonical ([ER16]). The lean principles that play an important role in the experience of this highly distributed agile team include end-to-end view of the process, removing bottlenecks in the process, and Kaizen where process improvement experiments are encouraged.

One of the lean principles, eliminating waste, is a recurred theme in [ER6, ER8, ER10, ER11]. In Wireless Data Services Global, a service provider to wireless companies and mobile phone manufacturers, the development teams "*have experienced tremendous positive effects from utilizing Extreme Programming practices on development teams*". However, they "*have yet to find the agile path to regularly providing positive business value*" ([ER6, p. 175]). Over the time, they have found a family of four agile practices that merged the XP principles of implementing the "highest value features first", and "don't do anything extra", with lean principles such as "eliminate waste" to address the highlighted issues. The four practices – Value-based Investment Decisions, High Confidence Stories First, Incremental Story Delivery, and Story Ownership – embody both agile and lean principles and are believed to be most effective when applied together. In [ER8], Sabre Airline Solutions, a company developing products for the airline industry, encountered agile plateau effect. In the effort of overcoming the plateau, the company chose to apply lean concepts, such as seeing waste and value chain mapping, to brainstorm the ways to improve quality, and develop quality improvement goals and action plans accordingly. [ER10] examines the use and adaptation of the "product owner team" practice in BT, British Telecom. Product owner team is used to manage the details of what should be built in a project implementing an up to 24 Mbps service over the 21CN network. The lean principle of eliminating waste and the seven types of waste help the company to "*break the silo mentality and simplify the delivery by reducing the work in progress and developing collaborative teams focused on customers' prioritised needs*". It is believed that the project was successful due to "*the collaborative efforts of the core team to eliminate waste – applying one of the core principles of lean*". Similarly, in a mission-critical commercial-off-the-shelf upgrade project described in [ER11], lean thinking, especially eliminating waste, is used as the guidance to the improvement of the manual testing process which is crucial to the project.

Kanban is another frequently used approach to improve agile processes. [ER19] reports the projects in the IT department of ASR Insurance, one of the top 3 insurance companies in the Netherlands. While most projects that used Scrum were successful, other Scrum teams were having some difficulties. The major reason was that these projects most of the time were involved in operations work or small maintenance. The work was hard to distribute properly over sprints and often needed to be changed more frequently than the 2-week sprints allowed. What the client wanted was more flexibility and more control over the immediate results. The company wanted to keep the agile mindset and at the same time do something more appropriate for maintenance and operations so that they too can cooperate with the rest of the IT departments and projects. Kanban technique was adopted, together with the underlying principles – make work visible, limit work in progress and help work to flow. As a result, much better understanding and cooperation between developers from different technologies as well as with the testers was observed, even though the team encountered different team and organisational challenges.

A similar story is recounted in [ER23]. Fundamo is a provider in mobile financial services products. The Product and Technology teams faced heavy support tasks for multiple customers. Scrum was used for the development work, while kanban was used for support. The teams started with a simple basic kanban board and allowed the board to emerge and add complexity when necessary since "*a less formal process and kanban board, less reluctance to change*". However they did use a CFD (Cumulative Flow Diagram) from day one and recorded the number of issues in each column at the same time each day after daily meeting. This diagram helped measure cycle time and make predictions around their defect fix rate. They also

tried to achieve a better balance between demand and throughput by closely collaborating with Professional Service teams at weekly prioritisation meetings where they reviewed the current outstanding issues and agreed on what they would tackle next. The meeting drove the issues that they added to the kanban board. At the end, the kanban board advanced significantly with more columns, buffer areas and WIP limits. Kanban allowed for issues to be added to the work queue at any time rather than on a fixed cadence as in Scrum, and for releases to be decoupled from the sprint cadence so the teams have flexibility to release a patch at any point rather than at the end of the fixed-length sprint. Kanban also provided a mechanism to quantitatively measure the effects of any changes in the process so that they are able to quantify both their demand and throughput and ensure these stay balanced by changing explicit policies.

Metrics were also carefully implemented in [ER27] where the application of lean helped to better manage defect resolution process of SumTotal Des Moines, a Learning Management System provider that uses a Software as a Service (SaaS) approach to delivering its product to customers. The company has various issues with their defect resolution process. Defects were not getting resolved for a long time which upset the customers. The process did not lend to efficient use of expensive developer time of the production support team. With the help of the customer support team, the production support team was able to expose waste by creating a value stream map of the old process. The exercise revealed that a defect took roughly 4 days to fix but spent 100 days waiting. To address these issues, a new main queue was used, which was a First-In-First-Out (FIFO) queue. A process to keep track of each defect as it flowed through the development process was established:

> "*Each time a new defect was picked from the new FIFO queue, a developer pair was responsible for creating a card and placing it on the board within the In Progress column. As defects flowed out of In Progress into QA, the pair would move the card. Each day after stand up the team lead would take count of the number of defects in each category and mark them in a spreadsheet*." ([ER27], p. 271)

In this process, the most important metric to collect was current lead time; e.g. how long until a defect is fixed, which provided predictability to the customer support team. Each day the team lead would count the number of defects in six categories: Backlog, In Progress, Blocked, In QA, Release Ready, and In Production. These numbers helped the production team determine their cycle times and lead times for defects while their CFD helped them visualise their defect flow.

Technical debt is a serious issue faced by many agile teams, which seems to accumulate with the progress of iterations. [ER26] shows how a very large Fortune 200 financial services company, who had some successful pilots using agile and was in the midst of a company roll out, started to address this issue by making technical debt visible through so called "code Christmas trees" located in the walkway of the company. It evoked conversations and discussions among developers and whoever is interested. To avoid punitive uses of the information revealed by the code Christmas trees, root cause analysis techniques like fish bone diagrams and 5 whys analysis were applied. The teams never stopped with a single individual when searching for the cause of problems. Meanwhile, the company kept experimenting with new adaptations of their trees since it was believed that "*it is easy to overlook big, visible problems over time when they do not change materially*."

Continuous improvement was the central theme of [ER22]. FFM (Field Force Management) project of one of the biggest international telecom companies had been facing several challenges, including no scope defined, no measurement of the team capability, "push scheduling" mentality was pervasive, the trust of business stakeholders in the team was minimal, and most team members were junior with little experience and exposure to agile methods. While the team started to use agile requirements analysis, they set up a kanban board to track each story as it flowed through the work flow. From the very beginning, they instilled the "Pull Scheduling" concept into the team, which managed the queue of items that should happen next; the team usually pulled an item off the top of the queue when planning work. The team pulled stories from the list in each iteration based on team capacity and capability. Continuous improvement opportunities were identified through the application of value stream mapping, which revealed that there was too much waste in the deployment process flow. Team dependences, or segregation of cross functional teams, were identified as a cause. With team dependencies, it would be very difficult to deliver even a small feature set since it required a large amount of communication and coordination among teams. The team had also learnt the importance to pull the "Andon" cord to "stop-the-line" if a true iterative iteration (in which the development team produce a releasable application) could not be achieved.

Continuous improvement was also at the heart of Systematic's experience. The result of the pilots in Systematic reported in [ER9] confirmed the general idea of using lean mindset as source for identification of new improvements. Actually that was what happened in Systematic later on, which is the main subject of another two experience reports about the same company ([ER15, ER25]). Viewing lean as a "*Scrum Troubleshooter*", Systematic used PDCA cycle with an A3 problem solving technique and the 5 why root cause analysis, to identify problems and opportunities for organisational wide improvement. Senior management involvement is significant in this case:

> "*Many of the adjustments implemented are characterized by being desired but beyond the control of the projects. Had the projects driven these improvements on their own, they might have discarded the ideas for adjustments because of the need to involve senior management or VP's in the decision. When the problem solving is initiated by senior management, the negative impact of the problem is viewed both from the project perspective and also from the business unit or company perspective. In this larger problems build high management larger perspective commitment, because the impact of the problem and the benefit of the solution is visible in a larger scale*." ([ER25], p. 174)

However, Systematic's most important learning from the improvements during the past five years was the lean concept of Jidoka – respect that those who do a particular part of work, are those who are best qualified to improve how this work is done. Make people responsible for solving their own problems and ensure that management supports it.

As shown in these 13 cases, continuous improvement, or Kaizen, is at the core of this type of lean application. Agile methods also advocate continuous improvement, but do not answer how it can be implemented. Lean approaches, instead, offer specific directions (eliminating waste in software development processes, focusing on flow, etc.) and specific practices (kanban, value stream mapping, root cause analysis, etc.) to improve agile processes continuously.

5.5. From agile to lean

Four experience reports contain the lean application cases that can be classified in this category. They are published recently (within 2009–2011). Table 10 is the profiles of these cases.

Yahoo! runs websites visited by hundreds of millions of users a day. Hundreds of development teams at Yahoo! rely on one another for code and services. Many of these teams have been using Scrum, including Yahoo! Sports team ([ER29]). Yahoo! Sports team used a lot of code written by other teams. Team interdependency was a commonplace. It was said that "*the worst customer for a Scrum*

Table 10
The cases in the "from agile to lean" category.

Experience report no.	Organisation name	Business domain	Lean elements applied
[ER29]	Yahoo!	Internet services	*Concept*: Flow *Principle*: Eliminate waste, Reduce batch size *Practice*: MMF
[ER18]	Inkubook	Online photobook	*Concept*: Pull *Practice*: Kanban board, Limited WIP, MMF
[ER20]	Codeweavers	Financial and insurance web services	*Concept*: Flow *Practice*: Value stream mapping, Kanban board, Limited WIP, CONWIP, MMF
[ER21]	(*not revealed*)	(*not revealed*)	*Concept*: Value *Principle*: Manage flow, Limit WIP, Visualise the workflow *Practice*: Kaizen, Visualise all work items, *Same-size work items*[a]

[a] Considered lean elements in the report but not included in our definition.

team is another Scrum team". Different Scrum teams had different sprint lengths, velocities and priorities. Often Scrum teams did not want to be distracted during their sprint. However, sometimes a sporting event would occur before another team could commit to resolving a dependency, which forced the Sports team to find a solution themselves. Due to the intense release schedule, the high demands of the product, and the many dependencies on other teams, the Sports team realised that they could not operate as usual. Among the transformational events that enabled the team to keep pace with the demands of the online sports world while maintaining team cohesion, motivation and high quality, an important one was daily release. Before the 2010 World Cup, the Sports team was deploying large feature releases that occurred at least every 2 week sprint. However, as the World cup event was set to begin, it became apparent that there would always be something that had to be delivered daily. What enabled the team to achieve such an aggressive cadence, among other themes, was the concentration of the team efforts on two lean elements: a better flow and smaller pieces of code. The team reviewed the timing of specific handoffs between teams everyday and made changes frequently to reduce the waste. They also focused on build automation as well as test automation by having the developers and testers share test development responsibility and do their work in parallel. Over time the team got better at identifying the MMF (Minimal Marketable Feature) and delivering smaller increments of code to production to a small subset of users.

Inkubook is an emerging player in the online photobook industry. The journey of the Inkubook team to improve their software development process is documented in [ER18]. The team initially adopted Scrum "by the book". However, they entered into a chaotic no process stage when the mandate arrived that a product would be delivered in sixty days. When the schedule slowed down again, in order to avoid burnout and staff turnover, the team moved back to Scrum, but this time it was really a "*flow-based, iterationless version*" of Scrum. After three months of being in the same "sprint", the team recognised that flow was working well for getting things done and therefore they abandoned estimates, implicitly organizing around a WIP limit of two MMFs. Finally, after several more months passed, the team accepted that they were using a work-limited, pull-based kanban approach and updated the usage and terminology to reflect that fact. According to the team, "*the use of a kanban implementation survived a round of layoffs, an extreme change in team and management composition, and is still being used today*".

A similar case is reported in [ER20]. The title "*From Chaos to Kanban, via Scrum*" illustrates the evolutionary path of the software development team in Codeweavers. The company is a UK business of approximately 20 people, delivering motor finance and insurance web services. The team comprises 8–10 co-located developers. Using simple inspect-and-adapt cycle to adopt one practice a time, the team adopted Scrum practices first to tackle the chaos, then used value stream mapping and adopted kanban board to have a better visibility of upstream and downstream activities other than work in progress on the story board. Step by step following the kanban adoption, the team introduced limit to WIP, fine tuned it while the team began batching tasks into MMFs. Along the way, the team also adopted automated regression test and adapted stand-up meeting to hold it twice per day to ensure tasks were worked on as things developed. At the end, the development process assembles more a flow-like lean process rather than a time-boxed Scrum process. Meanwhile, as the focus on flow and throughput became more deeply ingrained in the development team, the developers' view of the value stream gradually increased, from focusing only on the "in development" column to downstream to ensure the code developed was accepted and deployed to customers. Later still they looked further down-stream, helping customers to adopt the new services, and upstream, helping the business to decide what was needed and how to prioritise it. The focus was expanding even further into the company's sales and marketing functions.

If the transition of Yahoo! Sports team, Inkubook and Codeweavers from agile to lean was an incremental or unplanned process, the experience documented in [ER21] was a more systemic move. The software development team (not named in the report) had run a Scrum-based development process for several years. The practices included continuous integration, automated, nightly build and deployment to QA servers, and suites of automated unit and integration tests. In spite of the processes and practices in place, the team was still challenged by the issues such as frequent mid-iteration changes and non correlated work items. That is the reason why the team decided to embark on a lean journey. After careful research, the team arrived at a set of core concepts for their flow-based development: schedule individual value-adding work items, define a workflow, limit work in process, same-size work items, establish holistic key performance indicators, visualise all the work and the entire workflow, and improve relentlessly. The team found that "*it was fully possible to run agile software development without time-boxes by using a continuously updated work item priority queue instead*". The team's experience shows that WIP limit is a good control variable compared to controlling capacity and scope of work under high variability, and a single WIP limit (CONWIP) works well for the team.

The four cases in this category demonstrate that a comprehensive application of lean approaches can transform agile processes. Starting with the application of lean techniques to improve agile processes, these organisations ended up in a situation in which lean processes become dominant and agile practices play the supporting role.

5.6. Synchronising agile and lean

[ER30] reports an interesting case of one agile team and one kanban team working in parallel in a synchronised manner to address different aspects of the same development process. Much like most systems development teams, the teams in WMS, specialised in interactive entertainment, were constantly working to balance large scale system feature upgrades that make up about 60% of the work with small-scale changes and bugs that cover the remainder of developers' time. Over the time two coordinated agile/kanban teams were formed. The agile team run in a typical agile fashion. They were responsible for all large-scale projects that affect the system including any architectural roadmap work that may lead to structural changes to the software as a whole. The kanban team instead were responsible for all small feature requests along with bugs. The team used a kanban board to manage the development process only, starting from the currently selected top 5 priority items and spanning all the way to a deployed queue. They maintained a cycle time and lead time metric integrated with the story point system of the company. WMS's experience shows that:

> "*. . . running both [agile and kanban] processes in synchronization, and not blending them, has been highly valuable for our organization... The random interruptions and fire fighting of supporting a product in the field always seems to cause disruptions to Iterative teams. By adding a synchronized Kanban team alongside an Iterative team we have been able to even out our iterations and create a productive and healthy work environment where we are able to meet our customers' needs.*" ([ER30], p. 268)

6. Discussion

As shown in the findings section, there can be various ways to apply different lean concepts, principles and practices in agile software development. Apart from one experience report that describes non-purposeful combination of agile and lean practices, 29 out of 30 experience reports demonstrate the application of lean approaches in agile software development with specific objectives and strategies. The five strategies identified are: agile within, lean out-reach; lean facilitating agile adoption; lean within agile; from agile to lean; and synchronising agile and lean. With the first two and the last strategies, even though lean approaches are used to facilitate agile process implementation and effective use within larger organisational contexts, agile processes remain relatively "pure" or intact. In contrast, the strategies of "lean within agile" and "from agile to lean" result in processes that blend both agile and lean elements, to various degrees.

Among these five strategies, the most common one is where different lean elements are used to improve existing agile software development processes, evidenced by 13 out of the 30 selected experience reports. As claimed in Poppendieck and Poppendieck (2003), lean concepts and principles have often been used as thinking or sense making tools to guide the practice of agile software development. The commonly cited lean concept and principle in these reports are value and eliminate waste. It can be argued that these two lean elements provide both target and route for continuous agile process improvement. Another discernible pattern across these 13 experience reports is that, in the past, lean had been used more as a thinking tool but in a less conscious manner; whereas in more recent years, the trend has been to adopt more and more concrete lean practices, as a conscious choice of the adopting organisations.

The kanban approach is also at the core of the strategy "from agile to lean". 3 out of the 4 experience reports in this category can be seen as detailed demonstrations of how kanban transforms agile processes. Although the kanban approach shares similarities to agile approaches, such as a prioritised feature list, it's primary concern is to limit work in progress (WIP). However, there is a second significant difference between it and agile methods. The concept of a time-boxed (fixed duration) iteration is no longer used. Instead, the kanban board is used to set clearly visible limits to the number of tasks allowed to be in progress at any given time. There is no fixed number for each WIP limit but by measuring the lead time of individual tasks, the WIP limits and process itself can be optimised (Kniberg and Skarin, 2010). Kanban software development pursues the concept of continuous flow or what (Hiranabe, 2008) refers to as "Sustaining Kanban".

What has been seen through the experience reports analysis is that the kanban approach suits particularly well to software maintenance or support type activities where uncertainty is higher than in normal development activities and change is more frequent than that allowed by agile iterations. Another point worth making is that, when an agile adopting organisation is mature enough in using agile and especially lean practices, they have a tendency to move away from time-boxed agile processes to more flow-based lean processes, as several experience reports have shown ([ER18, ER20, ER21]). It is also interesting to note that these experience reports were recently published (2009 and 2010). This, together with recently practitioners-authored kanban books/articles, might indicate that moving from time-boxed agile processes to flow-based lean processes is a recent tendency in agile software development.

It is also worth noting that the lean concepts, principles and especially lean practices applied in the 30 experience reports are only a subset of what are defined in the relevant literature (as described in Section 2). Some lean elements, such as the principle of waste elimination and the practicing of kanban, were found to be more often applied than others. It might be an indication that these elements complement agile processes therefore are a better fit within an agile context.

Last but not least, even though the six types of lean application were presented as distinctive, unrelated categories, and each experience report was classified under one category only according to its primary focus, it does not mean that organisations can or do use lean in one manner only. The case of Systematic, the subject of three experience reports ([ER9] in 2007, [ER15] in 2009 and [ER25] in 2011) is a good example of an organisation applying lean in different manners to suit different needs at different stages of agile maturity. Firstly the company used lean as a facilitating tool for a smooth Scrum adoption, then applied a lean mindset and analysis techniques as sources for continuous agile process improvement.

7. Conclusion

The recent focus shift from agile methods such as XP and Scrum to a lean approach in software development has been noticeable and evokes the interests of both research and practice alike. The objective of our study has been to investigate how lean approaches have been applied in agile software development. To explore this phenomenon, we have conducted a secondary data analysis of 30 experience reports containing real-world experiences of the application of lean approaches in agile software development. We have identified six types of lean application in these experience reports and categorised them in a more systemic way.

The findings of the study enrich our understanding of how lean can be applied in agile software development. Since the research on the broad topic of lean software development is considered a nascent area (Dingsøyr et al., 2008), we believe our study is an important addition to this branch of research in general, and on the topic of combining agile to lean in specific. The definition of lean approaches in software development in terms of lean concepts, principles and practices brings certain extent of clarity to

the understanding of lean software development. It contributes to a better conceptual basis for further studies on lean software development. The lean application types we identified can serve as a thematic map for the researchers who intend to conduct more in-depth study of the phenomenon of what is termed as "leagile software development" in this paper. A significant area for further research could be to provide operational guidance to help developers (i) map the various potential "leagile processes" to their own context and (ii) implement the selected process combination.

The practical implication of our study is that it reveals different strategies to apply lean concepts, principles and practices in agile software development. There is no one-type-fits-all solution. Each organisation should reflect on its own development context, project objectives, and constraints as well as reflect on the various aspects of agile and lean before embarking on the journey of "leagile software development". The potential strategies summarised in this study could provide them with some promising directions to explore. However, how to effectively tailor these strategies to suit the specific situation and needs of the organisation is a challenge yet to be addressed and therefore worth further studying.

One limitation of the research is that, since "agile" and "lean" are very broad and often poorly defined terms, their interpretations in the experience reports are often vague and the terms can mean different things in different reports. Future research could examine agile and lean at the more detailed and operational practice level, comparing the commonality and contradiction of agile versus lean practices. A detailed analysis of practice level comparisons was not possible in this study as not all experience reports provided the required level of detail.

Another limitation of our study is related to the secondary data analysis method we applied, which leads to the weak basis of the data source. It must be noted that researchers using secondary data must be aware of issues related to data quality and accuracy. Since the collected experience reports represented secondary data, we had no control on the quality of data and especially the level of details we desired. The 30 experience reports included in the analysis therefore were not equally informative and illuminative. In addition, the organisations covered in these reports vary in terms of sizes and business domains they operate, which may pose potential threat to the validity of the findings reported in this study. One potential avenue for future research would be to conduct primary case studies which allow to design a better sampling strategy, to collect more specific and in-depth data, and to explore the issues, challenges and opportunities associated with the application of lean in agile software development.

Appendix A. The list of experience reports included in the secondary data analysis

[ER1] Hodgetts, P., 2004. Refactoring the development process: experiences with the incremental adoption of agile practices. In: Agile Development Conference, pp. 106–113.
[ER2] Rand, C., Eckfeldt, B., 2004. Aligning strategic planning with agile development: extending agile thinking to business improvement. In: Agile Development Conference, pp. 78–82.
[ER3] Parnell-Klabo, E., 2006. Introducing lean principles with agile practices at a fortune 500 company. In: Agile 2006 (Agile'06), pp. 232–242.
[ER4] Poon, D., 2006. A self funding agile transformation. In: Agile 2006 (Agile'06), pp. 342–350.
[ER5] Weyrauch, K., 2006. What are we arguing about? A framework for defining agile in our organization. In: Agile 2006 (Agile'06), pp. 213–220.
[ER6] Yap, M., 2006. Value based extreme programming. Agile 2006 (Agile'06), pp. 175–184.
[ER7] Fry, C., Greene, S., 2007. Large scale agile transformation in an on-demand world. In: Agile 2007 (Agile 2007), pp. 136–142.
[ER8] Packlick, J., 2007. The agile maturity map a goal oriented approach to agile improvement. In: Agile 2007 (Agile 2007), pp. 266–271.
[ER9] Sutherland, J., Jakobsen, C.R., Johnson, K., 2007. Scrum and CMMI Level 5: the magic potion for code warriors. In: Agile 2007 (Agile 2007), pp. 272–278.
[ER10] Croix, A.D.S., Easton, A., 2008. The product owner team. In: Agile 2008 Conference, pp. 274–279.
[ER11] Geras, A., 2008. Leading manual test efforts with agile methods. In: Agile 2008 Conference, pp. 245–251.
[ER12] Goos, J., Melisse, A., 2008. An ericsson example of enterprise class agility. In: Agile 2008 Conference, pp. 154–159.
[ER13] Kinoshita, F., 2008. Practices of an agile team. In: Agile 2008 Conference, pp. 373–377.
[ER14] Thomas, J.C., Baker, S.W., 2008. Establishing an agile portfolio to align IT investments with business needs. In: Agile 2008 Conference, pp. 252–258.
[ER15] Jakobsen, Carsten Ruseng, Sutherland, Jeff, 2009. Scrum and CMMI going from good to great. In: Agile 2009 Conference, pp. 333–337.
[ER16] Lacoste, F.J., 2009. Killing the gatekeeper: introducing a continuous integration system. In: Agile 2009 Conference, pp. 387–392.
[ER17] Shinkle, C.M., 2009. Applying the Dreyfus model of skill acquisition to the adoption of kanban systems at software engineering professionals (SEP). In: Agile 2009 Conference, pp. 186–191.
[ER18] Willeke, E.R., 2009. The inkubook experience: a tale of five processes. In: Agile 2009 Conference, pp. 156–161.
[ER19] Maassen, O., Sonnevelt, J., 2010. Kanban at an Insurance Company (are you sure?). In: Proceedings of the 11th International Conference on Agile Software Development (XP2010). Springer Verlag, Trondheim, pp. 297–306.
[ER20] Rutherford, K., et al., 2010. From Chaos to Kanban, via Scrum. In: Proceedings of the 11th International Conference on Agile Software Development (XP2010). Springer Verlag, Trondheim, pp. 344–352.
[ER21] Birkeland, J.O., 2010. From a timebox tangle to a more flexible flow. In: Proceedings of the 11th International Conference on Agile Software Development (XP2010). Springer Verlag, Trondheim, Norway, pp. 325–334.
[ER22] Zang, J.J., 2011. A never ending battle for continuous improvement. In: Proceedings of the 12th International Conference on Agile Processes in Software Engineering and Extreme Programming, pp. 282–289.
[ER23] Greaves, K., 2011. Taming the customer support queue. In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 154–160.
[ER24] Wijewardena, T., 2011. Do you dare to ask your HR Manager to practice KANBAN? In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 161–167.
[ER25] Jakobsen, C.R., Poppendieck, T., 2011. Lean as a scrum troubleshooter. In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 168–174.
[ER26] Kaiser, M., Royse, G., 2011. Selling the investment to pay down technical debt the code christmas tree. In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 175–180.
[ER27] Prior, M., 2011. "You want to do what ?" Breaking the rules to increase customer satisfaction. In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 269–273.
[ER28] Smits, H., Rilliet, K., 2011. Agile experience report transition and complexity at cisco voice technology group. In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 274–278.

[ER29] Nottonson, K., 2011. Yahoo! Sports: sprint 100 & beyond. In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 252–255.
[ER30] Polk, R., 2011. Agile and kanban in coordination. In: Agile 2011 Conference. IEEE Comput. Soc., Salt Lake City, UT, pp. 263–268.

References

Ambler, S.W., 2009. Scaling agile software development through lean governance. In: Proceedings of Software Development Governance SDG'09 – ICSE'09 Workshop, Vancouver, Canada: IEEE Computer Society.
Ambler, S.W., Kroll, P., 2007. Best practices for lean development governance. Available from: http://www.ibm.com/developerworks/rational/library/jun07/kroll/ (accessed on 01.09.11).
Anderson, D., 2010. Kanban – Successful Evolutionary Change for Your Technology Business. Blue Hole Press.
Anderson, D.J., Garber, R., 2007. A Kanban System for Sustaining Engineering on Software Systems. Available from: http://www.agilemanagement.net/index.php/blog/A_Kanban_System_for_Sustaining_Engineering (accessed on 01.09.11).
Barton, B., 2009. All-out organizational scrum as an innovation value Chain. In: The 42nd Hawaii International Conference on System Sciences, Waikoloa.
Beck, K., 1999. Extreme Programming Explained. Addison Wesley, Reading, MA.
Boslaugh, S., 2007. An introduction to secondary data analysis. In: Secondary Data Sources for Public Health: A Practical Guide. Cambridge University Press.
Bradley, R., 2007. Push to Pull: How Lean Concepts Improve a Data Migration. AGILE, 13–17 August, pp. 365–370.
Charette, R.N., 2003. Challenging the fundamental notions of software development. Cutter Consort. Exe. Rep. 4 (6).
Coad, P., Palmer, S., 2002. Feature-Driven Development. Prentice Hall, Englewood Cliffs, NJ.
Cockburn, A., 2001. Crystal Clear: A Human-Powered Software Development Methodology for Small Teams. Addison-Wesley, Reading, MA.
Conboy, K., 2009. Agility from first principles: reconstructing the concept of agility in information systems development. Inform. Syst. Res. 20 (3), 329–354.
Dall'Agnol, M., Janes, A., Succi, G., Zaninotto, E., 2003. Lean management – a metaphor for extreme programming? In: Proceedings of the 4th International Conference XP2003, Genova, Italy, pp. 26–32.
Deming, W.E., 1986. Out of the Crisis. MIT Center for Advanced Engineering Study.
Dingsøyr, T., Dybå, T., Abrahamsson, P., 2008. A preliminary roadmap for empirical research on agile software development. In: Proceedings of Agile 2008 Conference, Toronto, pp. 83–94.
Dybå, T., Dingsøyr, T., 2009. What do we know about agile software development? IEEE Software 26 (5), 6–9.
Fagan, M.E., 1976. Design and code inspections to reduce errors in program development. IBM Syst. J. 15, 182–211.
Goldratt, E.M., 1992. The Goal: A Process of Ongoing Improvement. North River Press.
Goldratt, E.M., 1997. Critical Chain. Aldershot, Gower.
Hibbs, C., Jewett, S., Sullivan, M., 2009. The Art of Lean Software Development: A Practical and Incremental Approach. O'Reilly Media, Inc.
Highsmith, J., 2002. Agile Software Development Ecosystems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
Hiranabe, K., 2008. Kanban Applied to Software Development: from Agile to Lean. Available from: http://www.infoq.com/articles/hiranabe-lean-agile-kanban (accessed on 01.09.11).
Jalali, S., Wohlin, C.,2010. Agile practices in global software engineering – a systematic map. In: Proceedings of the 5th International Conference on Global Software Engineering (ICGSE 2010). IEEE Computer Society, pp. 45–54.
Joyce, M., Schechter, B., 2004. The lean enterprise – a management philosophy at lockheed martin. Defense Adv. Res. J.
Kniberg, H., Skarin, M., 2010. Kanban and Scrum-Making the Most of Both. InfoQ.
Ladas, C., 2009. Scrumban – Essays on Kanban Systems for Lean Software Development. Modus Cooperandi Press.
Liker, J., 2003. The Toyota Way. McGraw-Hill.
Liker, J.K., Hoseus, M., 2008. Toyota Culture: The Heart and Soul of the Toyota Way. McGraw-Hill, New York, NY, USA.
Middleton, P., 2001. Lean software development: two case studies. Software Qual. J. 9, 241–252.
Middleton, P., Flaxel, A., Cookson, A., 2005. Lean Software Management Case Study: Timberline Inc. Extreme Programming and Agile Processes in Software Engineering.
Morien, R., 2005. Agile management and the Toyota way for software project management. In: Proceedings of the 3rd IEEE International Conference on Industrial Informatics, (INDIN'05), Perth, Western Australia: IEEE Computer Society, pp. 516–522.
Mujtaba, S., Feldt, R., Petersen, K., 2010. Waste and lead time reduction in a software product customization process with value stream maps. In: Software Engineering Conference (ASWEC), 2010 21st Australian, 6–9 April, pp. 139–148.
Naylor, J., Naim, M., Danny, B., 1999. Leagility: integrating the lean and agile manufacturing paradigm in the total supply chain. Eng. Costs Product. Econ. 62 (1), 107–118.
Ohno, T., 1988. The Toyota Production System: Beyond Large Scale Production. Productivity Press, Portland, OR.
Perera, G.I.U.S., Fernando, M.S.D., 2007. Enhanced agile software development – hybrid paradigm with LEAN practice. In: Proceedings of 2nd International Conference on Industrial and Information Systems (ICIIS 2007), Peradeniya, Sri Lanka: IEEE Computer Society, pp. 239–244.
Poppendieck, M., Poppendieck, T., 2003. Lean Software Development: An Agile Toolkit. Addison Wesley Professional.
Poppendieck, M., Poppendieck, T., 2006. Implementing Lean Software Development from Concept to Cash. Addison Wesley Professional.
Raffo, D., Mehta, M., Anderson, D.J., Harmon, R., 2010. Integrating Lean principles with value based software engineering. In: Technology Management for Global Economic Growth (PICMET 2010), 18–22 July, pp. 1–10.
Reinertsen, D., 1997. Managing the Design Factory: The Product Developer's. Toolkit The Free Press.
Reinertsen, D.G., 2009. The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas Publishing.
Robinson, H., 1997. Using Poka-Yoke techniques for early defect detection. In: Sixth International Conference on Software Testing Analysis and Review.
Schwaber, K., Beedle, A., 2002. Agile Software Development with SCRUM. Prentice-Hall, Upper Saddle River, NJ.
Serignese, K., 2011. A Sprinkle of Agile: A Dash of Lean. SPTechWeb.
Smits, H., 2007. The impact of scaling on planning activities in an agile software development center. In: Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS'07), Waikoloa.
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. Int. J. Product. Res. 15, 553–564.
Thimbleby, H., 1988. Delaying commitment (programming strategy). Software IEEE 5, 78–86.
VersionOne, 2010. 5th Annual State of Agile Development Survey. Available from: http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf (accessed on 01.09.11).
Wang, X.,2011. The combination of agile and lean in software development: an experience report analysis. In: Proceedings of Agile 2011 Conference. Salt Lake City, UT. IEEE Comput. Soc., pp. 1–9.
Womack, J.P., Jones, D.T., 1996. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Simon & Schuster.
Womack, J., Jones, D., ROOS, D., 1990. The Machine that Changed the World. Rawson Associates, NY.

Xiaofeng Wang is a researcher in Free University of Bozen/Bolzano. Her research areas include software development process, methods, agile software development, and complex adaptive systems theory. Her doctoral study investigated the application of complex adaptive systems theory in the research of agile software development. Her publications include several journal and conference papers in major IS journal and conferences, including Information Systems Research (ISR), Journal of Information Technology (JIT), the International Conference on Information Systems (ICIS) and the European Conference on Information Systems (ECIS).

Kieran Conboy is an associate professor in Information Systems in UNSW, Australia. His doctoral research focused on agile methods for systems development as well as agility across other disciplines. Kieran's other research interests include systems analysis and management accounting in systems development projects. Some of his research has been published in various journals and conferences such as Information Systems Research (ISR), European Journal of Information Systems (EJIS), the International Conference on Information Systems (ICIS). Prior to joining NUI Galway, Kieran was a management consultant with Accenture, where he worked on a variety of projects across Europe and the US.

Oisin Cawley is currently a doctoral researcher with Lero – The Irish Software Engineering Research Centre at the University of Limerick in Ireland. Previously he had been working for 17 years in software development for mostly multinational companies, and has held many positions including Global Software Development and Application Support Manager. He holds a BSc in Computer Science from University College Dublin, Ireland and an MBA from Dublin City University, Ireland. He is a member of the Irish Computer Society.