Agile Methods in Embedded System Development: Multiple-Case Study of Three Industrial Cases

Kaisa Könnölä, Samuli Suomi, Tuomas Mäkilä, Tero Jokela, Ville Rantala, Teijo Lehtonen

PII: S0164-1212(16)30041-3
DOI: 10.1016/j.jss.2016.05.001
Reference: JSS 9748

To appear in: *The Journal of Systems & Software*

Received date: 17 March 2015
Revised date: 8 April 2016
Accepted date: 1 May 2016

Please cite this article as: Kaisa Könnölä, Samuli Suomi, Tuomas Mäkilä, Tero Jokela, Ville Rantala, Teijo Lehtonen, Agile Methods in Embedded System Development: Multiple-Case Study of Three Industrial Cases, *The Journal of Systems & Software* (2016), doi: 10.1016/j.jss.2016.05.001

Highlights

- Examines three cases on applying agile practices into embedded system development.
- Visibility of work and system-wide understanding increased.
- Improved communication diminished the need for internal documentation.
- Slow hardware development and specialization of team members challenged agile methods.
- If not possible to present working product, visualize the progress in other ways.

Agile Methods in Embedded System Development: Multiple-Case Study of Three Industrial Cases

Kaisa Könnölä (corresponding author), Samuli Suomi, Tuomas Mäkilä, Tero Jokela, Ville Rantala, Teijo Lehtonen

Technology Research Center, University of Turku, Turku, Finland
Email: { kmkkon, smsuom, tusuma, tetajo, vttran, tetale } @utu.fi
Web: http://embedded.utu.fi

Abstract

Agile methods are widely utilized in software development but their usage in embedded system development is often limited to software. A case study of three industrial cases was carried out to understand how to tailor agile methods effectively including also hardware development.

Agile practices, mostly derived from Scrum, were tailored to fit the needs of each team and the method development was closely followed. Surveys conducted in the beginning and in the end of the cases were compared and complemented with interviews to understand the new working methods and their effects.

Case evidence shows that interdependencies between work of each developer were taken into account better, visibility over the whole product increased and need for internal documentation diminished due to improved communication, but dividing hardware tasks into iterations was experienced difficult. With some tailoring, agile practices are beneficial also in the embedded system development.

To successfully adopt agile methods into embedded system development, the team must consist of all the project members, the natural cycle lengths of different disciplines and different knowledge between the developers must be accepted and built upon, and the progress of the product must be presented or visualized in the end of each iteration.

Keywords: embedded system, agile, agile method, case study

1. Introduction

Agile methods, such as Scrum and Extreme Programming, are widely used in software development. They emphasize customer collaboration, self-organizing teams, working software throughout the software development and welcoming changes in any phase of the development. Agile methods aim to make the development work more efficient and productive by improving the process flexibility and transparency using iterative and incremental development. In practice, the methods give guidelines on how to organize the teamwork according to the agile values [1].

Embedded systems are specialized computer systems that are designed for specific tasks and typically consist of software and hardware. The agile methods are still not widely utilized in the development of whole embedded systems and most of the present usage is focused on software development [2]. The hardware development, typically including electronics and mechanics, is left out of the scope of the agile methods. Enhancing teamwork and focusing on the essential could bring benefits also to embedded system development, but the special characteristics of combining hardware and software development must be taken into account.

In this paper, three cases of bringing agile practices into embedded system development are presented. Section 2 provides the background for the agile development and especially provides a view of the agile methods and their opportunities and challenges when utilized in the embedded system domain. Section 3 presents the case study method utilized in the three cases. The industrial cases, the practice definitions and adaptations in the companies especially from the embedded development point of view are presented in Section 4 . A survey on the ways of working was conducted before and after the method evolution, and the differences and similarities between these surveys, i.e. the effects of the adopted agile practices, are evaluated in Section 5 as well as the experiences of the teams based on the survey and interviews. Before the conclusions, in Section 6, recommendations for the adaptation of agile practices into embedded system development are given and the experienced benefits and drawbacks are reflected with software development.

2. Background

2.1. Agile Development

In 1970, Winston W. Royce introduced a method of dividing the software development process into two phases for small projects, analysis and coding, and into seven

consecutive phases for larger projects [3]. This straightforward model is currently referred to as the waterfall model. Even though Royce implied that iterations were needed when developing products with complex designs, the waterfall model formed a commonly used basis for software development for years. Alternative software development methods, such as iterative and incremental development, have also seen use during the decades [4] but they did not gain wider popularity before the introduction of agile software development in the turn of the century.

In the 1990s, some lightweight methods, such as Scrum [5] and Extreme Programming (XP) [6], emerged as alternatives to the traditional approaches driven by up-front planning. These methods emphasized small, co-located and self-organizing development teams working close to each other, taking advantage of frequent feedback gathered from close customer collaboration, and embracing change [7].

In 2001, a group of proponents of these lightweight methods formed "a Manifesto for Agile Software Development" (commonly referred to as the agile manifesto) which comprised four values and twelve principles listed in Table 1 and Table 2, respectively [1]. The values and principles of the manifesto encapsulate the common ideas in different lightweight development methods into a new concept of agile development. After the agile manifesto, the research on agile software development has been increasingly popular, especially after 2005 [8].

While the agile manifesto is written on quite an abstract level, multiple agile methods, such as the previously mentioned Scrum and XP, provide practical ways to achieve the goals behind the manifesto. Common to all of them are iterative and incremental development, close collaboration between all the stakeholders, welcoming changing requirements even late in the development, frequent delivery of working software, and trusting individuals in their work through self-organizing teams. When developing iteratively, i.e. in short cycles making it possible to change the direction, and incrementally, i.e. adding new working features on top of the current working product, the product development process proceeds step by step towards the final product in a controlled way without comprehensive plans being written at the beginning of the project.

While the presentation in the original manifesto is purely directed to software engineering, there has been interest to expand agile thoughts into other areas as well, for instance to management and product development [9]. From another perspective, methodologies and concepts that overlap with the agile manifesto had already been introduced before the manifesto was published. For instance, agile manufacturing, while lacking a coherent definition, is a collection of mostly high-level descriptions of competitive manufacturing environments where companies need to cope with irregular and unpredictable demand [10].

There are several recognized benefits of agile methods in software development. The agile methods can have positive effects on both productivity and wellbeing at work [11]. The iterative development process helps equalizing the burden during projects and eliminating the stress in the final parts of projects. There exists several surveys about the agile practitioners and their perceptions of benefits and drawbacks of agile development. The 10th Annual Agile Survey by VersionOne, a company developing agile lifecycle management software [12], lists the following top four benefits: 1) ability to manage changing priorities, 2) increased team productivity, 3) improved project visibility and 4) improved team morale/motivation. Similar effects were also recognized in a survey made in the Finnish software industry [13], where the top three benefits were 1) improved team communication, 2) enhanced ability to adapt to changes and 3) increased productivity. In a survey conducted at Microsoft [14] in several teams, the benefits were seen to be in improved communication and coordination as well as in the capacity to deliver releases quicker.

Also some challenges in applying agile methods in software development have been noted. According to the Agile Survey by VersionOne [12], the main barriers for further agile adoption were the inability to change the organizational culture and the general organizational resistance to change. In the survey conducted in Microsoft the main challenges were at scaling, and also too many meetings when utilizing Scrum were noted [14]. The survey in Finnish software industry revealed challenges in top management support, customer/supplier collaboration, and cultural change between development teams and the rest of the business [13].

Table 1: Agile values presented in the Agile Manifesto [1].

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions *over processes and tools*
Working software *over comprehensive documentation*
Customer collaboration *over contract negotiation*
Responding to change *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

2.2. Agile Development of Embedded Systems

To gather accurate information about agile methods in the embedded system industry, a systematic literature review was conducted by the research group of the authors in 2013 [2]. It was found out that agile development of embedded systems is not a completely novel field of academic research. Several papers have been published during the last few decades and many embedded system companies

Table 2: Agile principles presented in the Agile Manifesto [1].

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

are applying agile methods in one way or another. However, based on the literature review, most related work about the utilization of agile methods is centered around embedded software development while hardware is left out the scope of agile practices [15] [16] [17]. Additionally, it has been concluded that agile development can even be applied to mass-produced embedded systems where the full R&D process cannot be agile [18]. However, the hardware development is again left out of the scope in the described cases.

In a company where hardware development is done in-house or where the work consists entirely of hardware development, the question about the applicability of agile practices in hardware development cannot be dismissed. While there are more general-purpose methodologies (e.g. agile manufacturing) available that might be applicable in some types of embedded system development, guidelines on how to organize development work on a weekly basis do not exist for embedded system development in the same way as they do for software development (e.g. Scrum and XP).

Many of the principles of the agile manifesto, such as the principles concerning people and teamwork, can be directly transferred over to embedded system development while some of them require changes or reinterpretations. For instance, the manifesto declares that working software should be the primary measure of progress: in the embedded system development, the principle can be interpreted in a way that the primary measures for progress are actually demonstrations of the whole system being developed. [19]

2.2.1. Opportunities

The popularity of agile methods in software development is due to the benefits they bring. Agile methods have positive impact on the embedded system development, at least in the software domain [2] and can be expected to have opportunities also in the embedded system development. The main opportunities for the agile methods in embedded system development are mostly related to the discipline and the fluency of development work as well as the efficiency, productivity and flexibility.

System-wide understanding. Agile practices have been noted to have positive effects on communication inside the development teams [20] in software development. Better communication will enhance the system-wide understanding, which can be the most focal benefit for embedded system development. Agile practices may give tools for understanding better the interdependencies between the different sections of the system being developed and consequently align the development work better.

Managing changes. Static requirements and specifications are often unrealistic in many new product development projects and embedded system development is not an exception. The plans change during the project and the final product is in many ways different from the dominant impression at the beginning of the project. The agile methods help to handle the changing requirements in the projects where the specification evolves towards its final form iteratively throughout the project. Typically, development teams are forced to cope with the changing requirements by working overtime, for example. Agile practices accept the changing requirements as a convention and give tools to handle them [7].

Managing interdependencies fluently. An embedded system development team is typically a multidisciplinary team with different domains of expertise where there can be complex interdependencies between the work of different people. These kinds of teams may face situations where a developer has to wait for someone else to finish a task before another task can be proceeded with. These interdependencies cause critical paths where the fluency of the work is endangered. Agile practices attempt to improve the system-wide understanding and transparency in a way that these kind of situations could be solved.

2.2.2. Challenges and Obstacles

Opposite to the opportunities, the special domain of embedded system development presents also challenges to the core ideas of agile development. These need to be taken into account, when tailoring agile methods in embedded system development.

One change may affect the whole system. The interactions between hardware and software are sensitive to changes [15]. Any changes in the design can cause variations in timing or other behavior inside the system. While the agile methods accept the fact that changes will occur and try to postpone the decisions as late as possible, in hardware development these interactions tend to force at least some up-front design to be done [21] and hinder the usage of refactoring, a practice used for instance in XP [22]. Critical interfaces and characteristics of the developed system should be defined early enough in the process to give space for agile development in these given boundaries.

Delivering a new version of a product in short cycles is challenging or even impossible. In software development, it is possible to release small increments of working and tested software at the end of each iteration. In hardware development, the natural cycle of development is longer [23] consisting of various more or less systematic tasks, such as circuit board design, prototype manufacturing, testing and verification. Also the cost of manufacturing a new circuit is high and thus creating a new working product consisting of new hardware parts every few weeks is often impossible. These factors cause the inevitable waterfall-shaped process on some phases of the product development and require the system level documentation to be written [21].

Teams consist of specialists of different disciplines. Agile methods promote cross-functional teams where every member can complete any task and, rather than only being responsible for their personal work, is collectively responsible for the outcome of the whole product development project with other team members. In embedded system development, the individual team members are usually highly specialized in different professional tasks either in hardware or software development, creating natural barriers that are likely to prevent the circulation of tasks inside the whole development team. The software–electronics and electronics–mechanics interfaces, both technical and interpersonal, highlight the importance of transparency, collaboration, efficient documentation and communication between the different disciplines. In an organization, where agile practices are already successfully utilized, achieving cross-functionality might be possible in a way that personnel from different disciplines understand each other's work better and can therefore synchronize their work more efficiently. Complete cross-functionality, i.e. everyone being able to complete every task, would solve the problem, but is likely impractical, since it would require the team members to have multiple professions and specialization areas.

3. Case Study Design

While there are challenges and obstacles in the adoption of agile practices in the embedded system development, such as developers with different knowledge and difficulties in releasing a new version in every iteration, the core ideas behind the agile methods can offer benefits for embedded system development as well. If the obstacles can be avoided, the opportunities, such as better system-wide understanding and managing both changes and interdependencies, can be very beneficial in various embedded system development projects. Evidence for understanding the challenges, understanding the changes required to the agile methods and supporting the benefits of the adoption needs to be gathered. For gathering evidence, a case study provides a valuable opportunity to understand the effect of agile methods in embedded system development in real-life context.

3.1. Case Study Methodology

A case study can be defined as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident" [24]. Even though this definition originates from social sciences, it fits well for software engineering case studies, where many factors impact the outcome of a software engineering activity [25].

In software development, the case study methods have been defined mostly based on other research areas such as social sciences or medicine. The first recommendations for the use of case studies on software engineering were focused more on quantitative data in the mid-90s [26], but later on also qualitative data was taken into account at the end of the 90s [27]. From there the recommendations evolved for instance to guidelines or templates [28] [29]. One of the first papers in software engineering to report the utilization of case study methodology was published in 1988 by Curtis et al [30]. In the recent years also case study papers about agile software development based on the templates and recommendations have been published [31].

A case study aims to understand better how and why the engineering process works and to find a way to improve it. Following a case study protocol helps to ensure the quality of a case study, i.e. to take into account the theoretical basis including formulating research questions, using triangulation, presenting the chain of evidence with traceable reasons and arguments, fully documenting the case study and formally reporting the case study [24].

In our cases, a case study approach based on [25] is followed. The case study is both i) explanatory, i.e. seeking understanding over the effects of agile methods in embedded system development and ii) improving, i.e. trying to improve the development process.

The case study process, illustrated in Figure 1, can be divided into three phases, which partially overlap. The

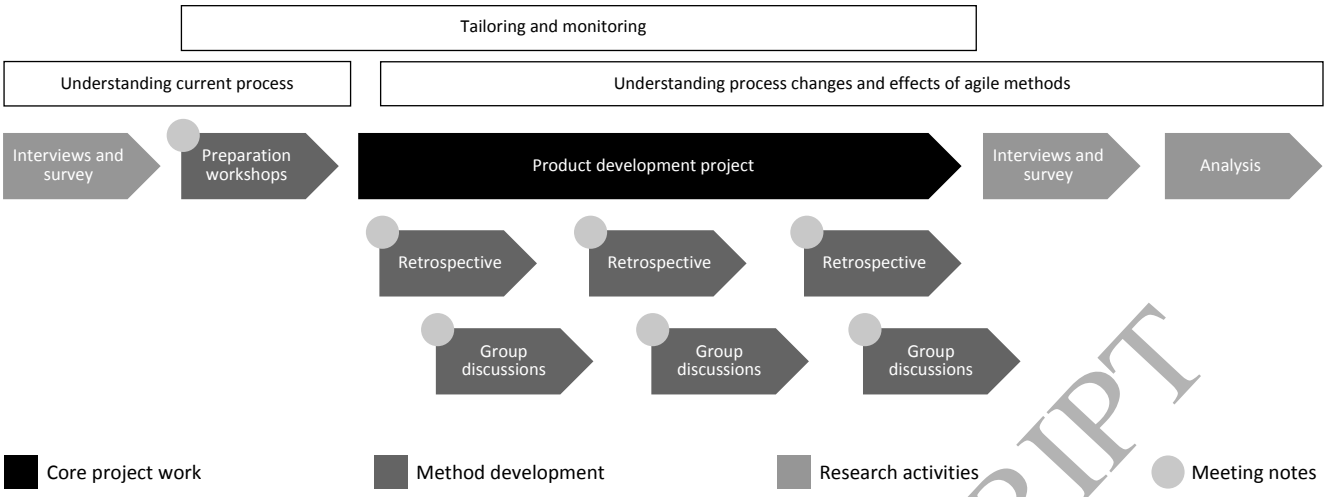

Figure 1: The case study process.

first phase is to understand the current situation inside the company. The second phase of the study consists of developing and tailoring agile practices to fit both the embedded system development and the company culture. The aim of the third phase is to gain knowledge on changes of the process through close follow-up in retrospectives and group discussions, and also to understand the effects of the changed process through a survey and interviews.

3.2. Research Design

The objective of the case study carried out on this work is to investigate how agile methods can be applied to embedded system development for the purpose of understanding what benefits and challenges agile methods have when utilized in embedded system development.

Since agile methods offer improvements in software development, the theory is that *agile methods can offer benefits to embedded system development similar to benefits in software development*. However, *the special characteristics of embedded system development need to be taken into account*. The most closely related theory from the existing literature [12] [13] [14] presents benefits and drawbacks connected to the use of agile methods in pure software development. Also, [15] [16] [17] suggest that there is need for modifications to agile methods in embedded context but analyze the situation only from the embedded software development point of view thus omitting the hardware development.

In order to gather more compelling evidence, multiple-case design was utilized [24]. In the three cases, the development process was changed to utilize agile development in order to find answers to the following two research questions:

- RQ1: How and why do agile methods need to be tailored in order to take advantage of their full potential in embedded system development?
- RQ2: What are the experienced benefits and drawbacks of agile methods in an embedded system development project and how they differ from software development?

The research questions are related to the case study setting so, that the second phase of the study, i.e. monitoring and tailoring agile methods focuses on the first research question. The third phase answers to the second question about the benefits and drawbacks the developers experienced.

3.3. Case and Subject Selection

Three cases provided a possibility to apply and monitor agile methods in a real-life environment in three companies that agreed to take part into the research project. The development of embedded systems in each of these three companies included also new hardware development, providing a possibility to understand better the special characteristics and challenges hardware development poses to agile methods. In each of the three companies one team was selected as a unit of observation.

Case A was conducted in a digital RFIC team of LM Ericsson, a multinational company that provides various devices and services in the field of communications technology. The company of case B, Nordic ID, develops and manufactures mobile and fixed RFID and barcode reader devices and services which can be used in retail stores and warehouses, for example. Case C was conducted in Nextfour Group, which develops embedded systems for medical, industrial and safety-critical markets based on their clients' requirements.

In the case projects, the outcome of each case project was either a whole embedded system or a part of it. In Nordic ID and Nextfour, the products usually consist of both software and hardware. In Nordic ID, software and

electronics development is done in-house, while the mechanics design and manufacturing are outsourced, whereas in Nextfour the design of software, electronics and mechanics is done in-house and only manufacturing of mechanics is outsourced. In Ericsson, the selected team was a hardware development team, specifically developing integrated circuits.

All three cases were conducted in Finland. In Nordic ID, the team was resided in two locations within approximately 50 km apart from each other, whereas in the other two companies the team resided in one location.

The characteristics for each case project is presented in Table 3. These three different case studies provided a representative collection of different kinds of teams developing embedded systems. An important difference between the teams was, that for case C, agile methods were somewhat familiar already, whereas for cases A and B they were introduced for the first time.

The selection of the project teams in the companies, where the new practices were piloted, was based mainly on availability and typicality of the project. In case A, the team was selected beforehand and the driving factor for the project selection was the availability of the project, i.e. what was planned for the team for that time period. Inside the large company, the team typically designs similar products, so the typicality was also ensured. In case B, besides availability, also size and typicality affected the selection of the case project: the desire of the company was that the case project would include a relevant portion of the R&D personnel and it would be a new product development instead of a customization for a special customer. In case C, the case project was desired to include both hardware and software development as was typical for the products of the company. The internal platform development was selected, since it enabled more experimenting than a customer project. The lack of a real customer was simulated by giving the CEO of the company the customer role for the project.

3.4. Data Collection and Analysis

Five different data collection methods were used to ensure data triangulation: company specific process documentation, semi-structured interviews, surveys, participant observation and focused group discussions. The survey produced quantitative data, whereas other collection methods were qualitative.

3.4.1. Understanding the Initial State

In the initial phase, there were three methods of data collection: documentation, interviews and a survey. The company specific process documentation gave the first insight on how the development process was organized inside the company and provided a basis for forming the interview guide.

To understand the influences of the process to the daily work of the employees, semi-structured interviews were conducted for different levels of employees from the developers to the team leader or the CEO depending on the size of the company. The interviews were mostly individual interviews, but some developers were interviewed together. The number of interviewees was 12 in case A, 8 in case B and 5 case C. There were always two interviewers: one of the interviewers took notes which were completed afterwards using a recording when needed. The utilized interview guides covered all the areas of the product development: requirements and specifications, communication within and outside the team, team work and work division inside the team, testing and its organization, development process and feedback. Also the challenges in the current process were discussed. From the completed notes, the data was rearranged into the areas of the interview guides and a report was created. This report was presented to the company to provide the understanding about the current process by the researchers.

A survey for the whole team or R&D department gave insight into whether the aspects found in the interviews were general or personal. This survey was created based on the knowledge gathered from the process documentation and the initial interviews as well as from general knowledge about the agile methods and the product development. The approximately 100 statements covered similar areas to the interviews and the areas with some example questions are presented in Table 4. The utilized format was a four-level Likert scale for agreement (strongly agree, agree, disagree, strongly disagree) or five-level Likert scale for frequency (never, seldom, sometimes, often, always or too seldom, somewhat too seldom, in ok intervals, somewhat too often, too often) or amount (too little, somewhat too little, adequately, somewhat too much, too much). In addition to the statements the survey contained open questions about the negative and positive sides of the work in terms of productivity, fluency and meaningfulness. The survey was conducted in Finnish.

The team of researchers distinguished from the survey data the most problematic issues: basically these were the statements with the most disagreements. For example in case B, two thirds of respondents somewhat disagreed with the statement *I know all the time what other developers in other teams of the project are currently working on* and only one third somewhat agreed. Taking into account the context, i.e. the teams being the software and hardware teams, this was seen as one of the improvement areas. These results were grouped together to themes, such as insufficient feedback, and for each theme one researcher was responsible for analyzing the interview data in order to find out whether a similar theme existed in the interviews. Then the researcher also gathered practices related to the theme from agile methods. After going through the themes and possible solutions to them with the research team, the themes and a first proposal of possible agile practices were presented to the company representatives.

Table 3: The case projects.

	Case A (Ericsson)	**Case B (Nordic ID)**	**Case C (Nextfour)**
Project	Upgrade work on older RFIC product design	Already started new RFID reader development project	Internal platform development
Team composition	Four hardware design groups	Hardware and software development distributed in two locations	Hardware and software development
Team size	14 (2–5 in each group and a team leader)	7 (3 software and 3 hardware developers, a project manager)	5 (2 hardware and 3 software developers; out of which 2 worked in this project only half-time)
Case study length	5 months	6 months	2 months
Case study schedule	Nov. 2013 – Mar. 2014	Oct. 2013 – Apr. 2014	Aug. 2013 – Oct. 2013
Previous agile knowledge	None	None	Some
Size of the company	Large	SME	SME

3.4.2. Method Tailoring in Case Projects

Before beginning to change the current practices, a workshop with researchers and company representatives was organized to select and refine the initial practices for the case project. While the suggestions came especially in the beginning from the researchers, it was a decision of the team which practices to utilize and how, as well as how to alter the practices during the case.

The data for the evolution of the practices was collected through participant observations and group discussions. The researchers participated into the meetings of the new process as observing participants taking notes on the process related events, e.g. who were present and how the practices were implemented. Especially retrospectives (see Table 5) offered a good insight into how the team changed the practices to better fit their needs, but the researchers also observed some of the other meetings. Group discussions with selected team members were also organized once in every iteration. The participants in the group discussions from the company side were the facilitators, i.e. a member or members of the team taking responsibility of promoting the new practices inside the team, and the team leader or the project manager. These discussions provided insight into what had happened between the observations and a possibility to discuss about how agile methods could help the team in their challenges.

The meeting minutes and observation notes of the researchers were organized into a pilot diary. After the case projects, the data in the diaries was organized into categories of different practices, containing information about how the practices evolved during the cases. From the practice categories, themes were specified: whether the selected practice or a modification in it was adopted directly from agile methods, tailored especially to fit the embedded systems development or tailored due to the company culture. The tailoring required by the embedded system domain is discussed in more detail in Section 4.

3.4.3. Understanding the Effects of Agile Practices

The end survey was conducted after the case project. In addition to the similar questions to the initial survey, the teams were also inquired about each new or altered practice as well as about the current practice set and its utilization. Interviews were conducted again to provide a way to understand how the team had experienced the new working methods and to understand what had happened after the case project ended. In these interviews the results of the surveys were discussed, too. The number of the interviewees in these interviews were 2 in case A, 5 in case B and 3 in case C.

The data collected was analyzed after the end survey. The initial and end survey were statistically compared, giving insight to the changes between the surveys. The noted changes were organized to themes and the end interview data was also organized according to these themes. The usability, usefulness and potentiality of the practices were evaluated to see if there was any practice which stood out from other practices. Open answers related to the practices were grouped according to each practice in order to see which themes were popular within each practice. All of the results of the analysis were discussed with the research group before presenting the results to the company in a meeting and in form of a report. The results and their analysis are presented in Section 5.1.

Table 4: The survey areas and example statements.

Covered area	Description	Example question/statement
Background	Work experience, current position, discipline.	How long have you worked in this company?
Documentation	Documentation sufficiency in terms of knowledge transfer, own usage, reuse, common working methods.	Documentation is sufficient in order to transfer knowledge between teams.
Communication and knowledge transfer	Availability of required knowledge of work done by others, general information availability, amount of meetings, confusion handling.	I know all the time what other developers in my team are working on.
Teamwork	Task distribution, possibility to affect own tasks, shared responsibility in the team towards common goals, team members supporting each others.	The work objectives are specified together in the team.
Requirements and specifications	Amount of change of requirements and reacting to them, possibility to affect specifications, modularity and reuse.	I can affect the specifications of the products.
Product development process	Process usefulness, process evolution from high level to details, division of the process to phases, risk management, process development.	The project goals are divided into clearly understandable phases.
Challenges at work	Schedules and possibility to affect the goals, work amount and change in it, handling multiple projects at same time.	It is difficult to estimate the amount of work for a specific time period.
Tools	Tool usage and usefulness.	Project tools (e.g. bug reporting systems, version control) make it easier to organize my own work.
Feedback	Individual and team feedback, learning from mistakes and successes.	Feedback received by the team is processed systematically.
Work efficiency	Work efficiency, quality and prioritization.	Prioritization of work is fluent and I get sufficient support for it if necessary.
Customer interface	Communication and collaboration with different customers (inside or outside the company, depending on the situation).	There exists a common understanding over the features and their prioritization between the client and the development team.
Testing	Testing and verification throughout the product development process.	Testing is taken into account already in the definition of the project.

3.5. Validity and Reliability

Having three different cases provides a possibility to compare the cases and especially to find out whether there is a difference between the company for which agile methods were already somewhat familiar and the companies for which they were completely new. On the other hand, if similarities between the companies were found, it would give the possibility to generalize the findings in order to give more understanding about the phenomena of agile embedded system development.

From the construct validity point of view, it is possible to discuss the used terms during the interviews in order to make sure that they are similarly understood by the interviewer and the interviewee. This is not possible in a survey. Thus it was seen important to return the results to each team in separate meetings to clarify the interpretation of the questions. The researchers were also often involved in the meetings of the companies during the cases and thus deepened the understanding on how the participants interpreted the terms in order to utilize them similarly.

There are always some changes occurring inside the companies during the cases regardless of the existence of the case study. It is possible, or even probable, that these changes also affect the responses in the surveys, causing a threat to the internal validity. The interviews in the end complemented the surveys, and at least some of the other changes than the agile methods in the companies were mentioned by the interviewees. For example, in case A, a new hour reporting tool for working hours was taken into use during the case project, and it affected the opinions about the project and process tool usage.

There were challenges to the external validity especially with the end survey, where the answer rate diminished from the initial survey and the number of respondents was quite small for a proper statistical analysis. The results were generalized as results of the whole team, when only 67% – 75 % of the employees answered to the survey. Even though there were three teams in three companies, both the company and the team culture affect the results from the generalization point of view. The results of the surveys were always discussed with the team in order to give the team a possibility to reason their answers to be able to remove the effect of the company culture.

Researcher bias is one threat to reliability, especially

with the qualitative data collection and analysis. Some countermeasures to researcher bias were taken: 1) in the interviews, there were always two researchers present, 2) in the meetings, there were often present more than one researcher, 3) the analysis was discussed with the whole research team and 4) all the reports delivered to the companies were revised by several researchers and discussed with company representatives.

Another threat to reliability is to consider only one point of view. Often the change of a process in a company is led by the management or the team leader, leaving the team only to follow and implement the decisions. To get a broader view, the interviews covered all the disciplines and roles of the team in order to gain understanding from several viewpoints.

From the reliability point of view, the researchers influenced the selection of practices by presenting a set of possible solutions. The responsibility of selecting and refining the practices was left to the team. This was done to maximize the benefits for the companies, which was also one of the objectives of the project.

4. Method Evolution in Case Projects

The adaptation of agile practices to embedded system development started with finding the similarities between software engineering and designing of electronics and mechanics, and examining which agile practices could be straightforwardly utilized or customized to fit into these areas. By starting from the more general practices, the influence of the practices can be seen and the need for more field specific practices recognized. Also, the current process and the challenges faced in each of the companies played an important role in the customization of the practices, and the agile values guided both the selection and the tailoring of the practices.

4.1. Understanding Initial State

In the initial state, the interviews and the survey revealed some challenges in each company, which motivated for changing the ways of working.

In case A, the team was originally divided into three groups with specific working areas. The groups acted as customers to each other: e.g. the output of one group was utilized by the other groups. The communication was based mostly on face-to-face communication. According to the survey and the interviews, three key areas of improvement were indicated: 1) fluency and planning of the work, 2) feedback enabling continuous improvement and 3) internal communication to create transparency. According to the survey, the amount of external disturbances, such as old projects or new requirements to the current project, was endangering the fluency of the work. Neither the team nor the individuals received adequate feedback. Also, the amount of documentation was seen inadequate and the team meetings were infrequent.

In case B, the product development process was plan-driven and based on milestones. From the developers' point of view, milestones were mainly seen as deadlines for documentation, and the actual structure of the development process was considered somewhat fuzzy. Basically the backbones of the development process were the highly experienced employees with intrinsic knowledge on what to do and when. The tools used and the documentation created differed between the hardware team and the software team, and these teams had difficulties in understanding each other's work and priorities. This had led to a point where the focus was more on the tasks at hand, instead of creating a product as a unified project team. According to the initial interviews and the survey, the identified four key areas of improvement were: 1) the transparency of the vision and motives behind the requirements, 2) the common tool usage and documentation practices, 3) the clarity of the development process, and 4) the experience of both team and individual work. From these four challenges the vision related challenge was left to the future projects, since the case project had been ongoing for a while and this challenge was related to the beginning of a project.

In case C, some agile practices had already been taken into use, but were not systematically utilized in all the projects. The used methods varied between projects based on the customer needs. The agile practices were used mostly in software development and taken mainly from the Scrum method. The developer teams were already quite self-organizing, and while many team members had expertise in different areas, the team was somewhat cross-functional. Some developers also took part in process development tasks in the company. Based on the interviews and the survey, the focus of the case project was decided to be 1) refining and formalizing the existing processes and 2) testing and feedback practices. The current development process was seen to be too heavy and no one was responsible for it. The testing practices were not seen to be systematic enough and the feedback was not received adequately.

4.2. Practice Evolution in Embedded System Development

In the beginning of the case projects, the researchers presented their ideas of the agile practices fitting each company and the teams selected and defined the details and practical implementations of the practices. Even though the presentations offered ideas from several agile methods, the initial method selected by the companies ended up having most elements from Scrum. Later on, the teams changed and refined the practices according to their needs.

Initial backbone of the new agile method. In general, in all the companies the backbone of the method was initiated by the researchers and ended up being similar: iterations, which started with planning of the work for the iteration and ended with reviewing the work done including also retrospecting how the method could be further improved.

The tasks under work were held in a backlog and the work during iteration was followed-up in status meetings. The two agile roles utilized in all the cases were a sponsor and a facilitator. The role of the sponsor was to provide support from the management level of the organization for the agile team. The facilitator supported the team from the inside to utilize the agile practices, similarly to a Scrum master in Scrum. Self-organization, i.e. giving the team a possibility to choose how they can best accomplish their work, was a key practice in all the cases, and the teams had a possibility to change also the process in a way they wanted. Especially in case C, self-organization was emphasized and discussed e.g. in every retrospective. The team defined details of each practice are collected in Table 5 including also short practice explanations.

The retrospectives, while not differing from software practices, offered a possibility to experiment and tailor the practices to better fit the needs of the project team and were thus an essential part of the practice evolution in all three cases. The retrospectives were mostly based on a new practice called the agile checklist, which was inspecting the selected practices, finding out what was good and bad and what required changes in them.

System-wide understanding. In agile software development, the team has both developers and testers, i.e. it consists of all the people related to the project. In the three cases, the scope of the team was widened already from the beginning to include both software and hardware developers, as well as testers. The definition of a team was redefined especially in cases A and B. In case A, earlier the team was splitted to four groups of their individual design roles, but now the work was planned as a whole team. In case B, earlier the team had composed of people of the same discipline, but now it was reformed to include all the members of the same project. The motivation behind this was to widen the understanding over the system and the work of other disciplines in order to manage the interdependencies. In the case C, the agile methods were already in use and the team already consisted of both hardware and software developers. The agile value of individuals and interactions was emphasized by making the team members collaborate more.

Integrating different disciplines. The slower nature of hardware development affected the practices. In case A, the status meetings were first held in three small groups, but after the first iteration the team decided to integrate them into one team meeting of fifteen people. At the same time, the frequency of the status meetings was diminished from 5 to 3 times in a two-week iteration. The team made the changes in order to enhance the information flow but also to ensure that the time is spent effectively in the meetings. In case B, the iteration length was decided to be tied to a natural development event (such as the end of layout design) in the beginning. This selection of iteration length led the first iteration to be 7 weeks. After the first iteration, the team changed the iteration length into exactly 4 weeks in order to be able to plan and execute the iterations properly – since it was a challenge in the 7 week iteration. In cases A and B the status meetings were held only once or twice a week instead of the daily standups of Scrum, since the team experienced that in one day the design evolves quite little, and thus discussing the work every day would have been a waste of time. These changes improved the interactions between individuals, as specified in the agile values, through making the team members collaborate regularly.

Reacting to changes. In case B, the availability and delivery time of components was an issue in some iterations. Agile methods made this issue more visible than before, making it possible to take it into account. Sometimes there were new features added to the backlog, or old ones dropped, when seen feasible. In all three cases, the short iteration was utilized as a checkpoint to see where the product was heading to and made it possible to change the direction when required.

In case A, there was a need to manage sudden new requests for the current product from other departments of the company, since they were interrupting the daily work. To address this, a spare planning meeting, a possibility to adjust the content of the ongoing iteration, could replace the status meeting in the middle of an iteration. The team utilized this practice several times.

The changes in the cases came more from the improved understanding about the product status during the development. Agile practices gave the possibility to react to these changes systematically when planning the work for the next iteration.

Managing interdependencies. The planning meeting was seen as a way to manage interdependencies between the hardware and the software development. In cases A and B, each individual or discipline had already been splitting the work into tasks before the planning meeting. The meeting focused more on aligning the work and going through the amount of work planned inside the iteration. In the case C, one of the team members was always responsible for each requirement, even though he/she could divide it into several tasks for several team members. Sometimes the planning meeting was divided into two parts: first the requirements were divided into tasks in groups of each discipline, and only then they were checked with the whole team in order to align the work. Both product and iteration backlogs were utilized by all three teams and they presented the work of the whole product team, not only the work of each discipline. Also the status meetings served as a place to check the schedule of the tasks which affected each other. The planning meeting differed from Scrum method: the team did not split the work into tasks together, but planned the work more within disciplines, and then aligned the work during the planning meeting.

Table 5: The practices utilized in the case projects.

Practice	**Description**	**Case A (Ericsson)**	**Case B (Nordic ID)**	**Case C (Nextfour)**
Iteration	A key feature of the agile approach. A project consists of a sequence of iterations, the length of which is pre-defined.	Length 2 weeks	Length 4 weeks	Length 2 weeks
Product backlog	List of features that are known to be necessary to finalize a product.	Custom spreadsheet based	Text and spreadsheet	Bug tracker tool
Iteration backlog	List of tasks that are known to be necessary to finalize an iteration.	Custom spreadsheet based	Bug tracker tool	Bug tracker tool
Iteration planning	A meeting for the team in the beginning of an iteration, where the goals and the functionality of the next iteration are decided.	Used	Held via teleconference	Used
Spare iteration planning	A meeting for the team that is arranged, if necessary, to enable the team to react to changes in requirements in the middle of an iteration.	Used a couple of times during the project	Not used	Not used
Review	A meeting at the end of each iteration, where the team and other stakeholders attend. A demonstration of the product is presented in the meeting and the work done in the iteration is inspected.	Used	Held face-to-face	Used
Retrospective	A meeting with the purpose to learn what works and what does not work in the current working methods and make adjustments for the next iteration.	Used	Held face-to-face	Used
Status meetings	Meetings during the iteration to keep track of the progress of the team and share knowledge of the tasks and challenges currently under work.	Held two to three times in an iteration	Held weekly using teleconference	Held every day
Maintenance hour	The last hour of each working day reserved for maintenance tasks. Used for reducing the interruptions to the current work.	Tested in the case project, but not continued	Not used	Not used
Agile roles	Roles utilized in the case projects.	Facilitator, sponsor	Facilitator, sponsor	Facilitator, sponsor, emphasized self-organizing teams

When the content of each iteration was planned together, it shifted the focus away from the documentation towards the product according to the agile values. Managing the interdependencies between the work of different developers was not tied only to the documentation and its readiness, but the tasks dependant of each other were agreed in the planning meeting and followed-up in the status meetings. This way the different parts of same feature were implemented simultaneously.

Working product. In contrast to software development, arranging the review meeting in embedded system development was difficult as there was not a working product to be presented in the end of every iteration. Instead the progress of the product and the work accomplished was reviewed. In cases A and B, the review meeting was a new practice. It focused on reviewing the tasks defined for the iterations and either closing them together or inspecting the statuses of the unfinished tasks. Even though this improved the knowledge of the progress of the product, it kept the focus on quite a detailed level. In case C, the team formalized a review template, which contained inspection of the main goal of the iteration, possible demonstration, going through the finished requirements against the definition of done, backlog prioritization and agreeing the goal for the next iteration with the customer. The demonstration was not obligatory and it was presented only when appropriate.

Utilization of backlogs was seen as a way to define the product at given time. Each team developed best practices for the implementation of the backlogs. In case C, there were three levels of backlogs: between the product backlog and the iteration backlog was a milestone backlog containing requirements which needed to be accomplished before a certain milestone. Also in case A, the need for release planning was noticed at the end of the case project, instead of only a long term product backlog and a short term iteration backlog. In case B, the same issue was tackled with the product backlog, where the requirements of each feature were specified as maturity steps to be developed inside the iterations.

In case A, maintenance requests interrupted the current project. To handle them, a practice called maintenance hour was introduced. As the last hour of every day, it was utilized to handle all the support requests received during the day. The idea of this arrangement was to reduce the interruptions in the current project due to the maintenance of previous products through categories: the request implementations were either postponed through the product backlog or implemented during the maintenance hour. As a rarely occurring exception, critical maintenance requests could be taken into execution immediately although affecting the current iteration. However, the team discarded maintenance hour and interruption practices after a short trial, partly due to other departments of the company not understanding or respecting the practices but also because the developers eventually did not consider them beneficial.

The reason for the trial of maintenance hour was to ensure that the focus would be kept in the current product without sudden interruptions. The utilization of review and backlogs shifted the focus from the plans to the product under development, as specified in the values of the agile manifesto. When the iteration content was agreed and checked together and the understanding of the progress of the product was more visible, there was a better understanding on how the individual tasks influenced the progress of the product. Still, the agile idea of having a working and deliverable product all the time required the most interpretation.

5. Results of Surveys and Interviews

In all the companies, the majority of the respondents of the initial and end survey were the same and this provided a possibility to compare the survey results through statistical methods. The answer rate decreased from the initial survey to the end survey in each case as presented in Table 6.

In addition to similar statements as in the initial survey, the end survey contained also a new collection of statements about the experiences of each individual practice and the utilization of the new methods. The survey analysis was complemented with interviews, a researcher group analysis, opinions shared when presenting the results, and answers to the open questions about the utilized agile practices.

5.1. Initial and End Survey Comparison

The quantitative analysis between the initial and end survey was done with a Wilcoxon signed rank test, which is a test of difference in location between two dependent samples [32]. To be able to utilize a test of dependent samples, only the respondents who took part in both surveys were taken into the statistical analysis (see Table 6). From the approximately 100 questions asked, here are presented the ones, where the statistical significance level (p-value) is below 0.05 according to convention. The significance level of each statement is presented also in the figures.

The analysis of the survey comparison was carried out in two parts, due to similarities in cases A and B, and their differences to case C: 1) the beforehand knowledge of the agile methods, 2) the percentage of the respondents to the survey taking part into the case project and 3) the time the end survey was conducted related to the case project and. In case A, the surveys were conducted inside the case project team, and in case B, all the survey respondents were from the R&D department, including only one person outside the case project team. In case C, only half of the respondents to the end survey took part in the case project while the other half consisted of other R&D department members. In both cases A and B, the end survey was conducted right after the case project, whereas in case C

Table 6: The survey data for the case projects.

	Initial survey answer rate	**End survey answer rate**	**Number of same respondents in surveys**
Case A	94 % (15/16)	71 % (12/17)	11
Case B	100 % (12/12)	67 % (8/12)	8
Case C	88 % (14/16)	75 % (12/16)	11

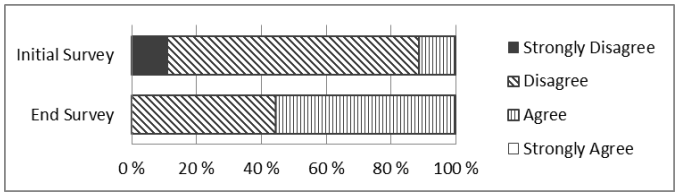

Figure 2: Case A: "Documentation is good enough for knowledge transfer between teams." (p=0.025)

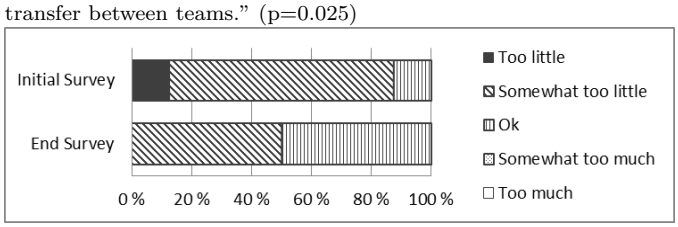

Figure 3: Case B: "The amount of documentation made by others for my own work is..." (p=0.046)

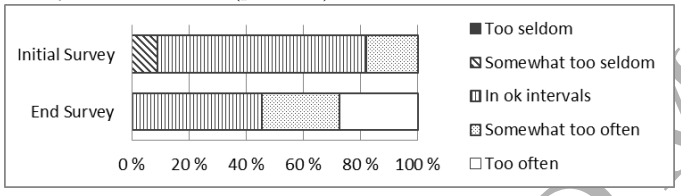

Figure 4: Case A: "Regular team meeting organization frequency is..." (p=0.033)

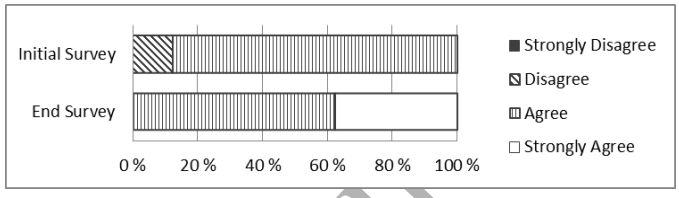

Figure 5: Case B: "The team members give their best expertise towards commonly specified goals." (p=0.046)

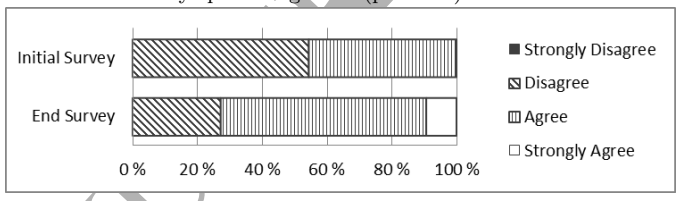

Figure 6: Case A: "Continuously evaluating and improving process feels like extra burden and takes time away from productive work." (p=0.046)

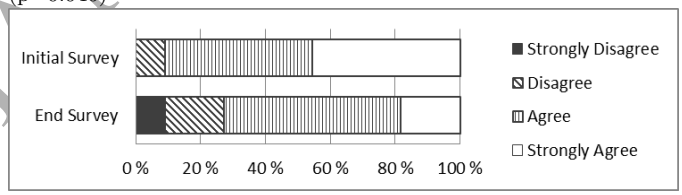

Figure 7: Case A: "The usage of project tools makes it easier to organize my own work." (p=0.014)

the end survey took place half a year after the case project had ended. This was due to the small size and short length of the case project, and especially the plan of continuing it later, which did not realize at least during the follow-up time of this research. The company of case C was the only one already utilizing agile methods – at least to some extent – before the case project, and the initial survey actually gave more positive results compared to the two other companies.

5.1.1. Cases A and B – Introducing and Defining Agile Practices

Some common themes could be found in the survey results in cases A and B: 1) diminished need for internal documentation, 2) improved visibility and teamwork and 3) challenges in changing the process.

Diminished Need for Internal Documentation. According to the end survey, the documentation related opinions had improved from the initial situation. Especially in case A, the documentation was better in terms of knowledge transfer between the teams as illustrated in Figure 2, whereas in case B according to Figure 3, the amount documentation made by other developers was felt more sufficient than before.

The documentation in both cases had remained similar to what it had been. The only change in documentation was that the product and iteration backlogs were taken into use and contained information in written form. According to the interviews, the new backlog tools and improved communication were the main reason for the diminished need for internal documentation.

Improved Visibility and Teamwork. Half of the respondents in case A considered that the amount of meetings was even too much as can be seen from Figure 4. The meeting frequency in case A did change quite dramatically, from one meeting in two weeks to two to three meetings per week. Even though the number of meetings also increased in case B – especially for the hardware developers – it was considered suitable.

The understanding about the work of other team mem-

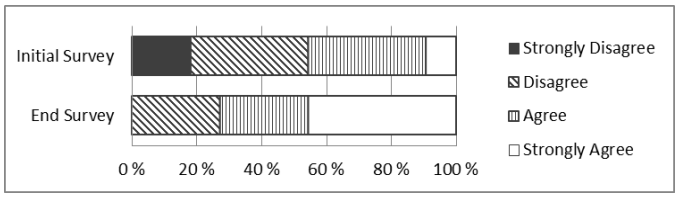

Figure 8: Case A: "Maintenance of old projects disturb the implementation of the new project." (p=0.011)

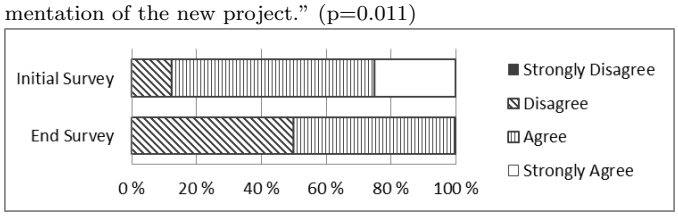

Figure 9: Case B: "My working was very efficient during the latest project." (p=0.025)

bers improved during the cases. According to Figure 5, the team of case B was contributing its full expertise to the commonly defined goals. The progress of the project was more tied to the practical tasks defined together.

Introducing the new practices involved adding new meetings and refining old ones. The new practices also involved developers to plan and share their work in progress with each other better than before. According to the interviews, these practices enhanced the internal communication of the team and visibility over the work in progress.

Challenges in Changing the Process. The developers were given a possibility to change the process. In case A, the improvement of the process was seen more as an extra burden as presented in Figure 6 and the project tools were not considered to ease the organization of the work of the developer as well as before as presented in Figure 7. The changes agreed together in the retrospectives were not actualized and made also the improvement of the process frustrating. Even though the new backlog tool was improved several times, it was not adequately used by all the team members. At the same time, also a new reporting tool for working hours was taken into use inside the company. These were seen to be the reasons behind the more negative attitude towards the project tools. As presented in Figure 8, in case A, the maintenance tasks disturbed even more than before. According to the interviews, this was due to the project situation, but it still indicates that maintenance is an issue to be addressed in the future.

The improved visibility had its downsides, too: Figure 9 shows that in case B the work was not felt as productive as before. Periodically reviewing the implemented work over plans visualized that all the planned work was not implemented causing the feeling of inefficiency. Still, according to the interviews, the transparency received was considered beneficial: through planning the work and defining the tasks together, the developers got a better understanding over the work of other team members.

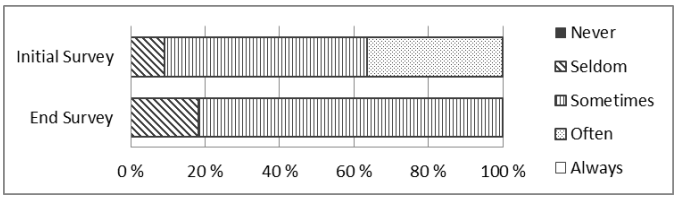

Figure 10: Case C: "It is difficult to evaluate the amount of required work for a specified time." (p=0.025)

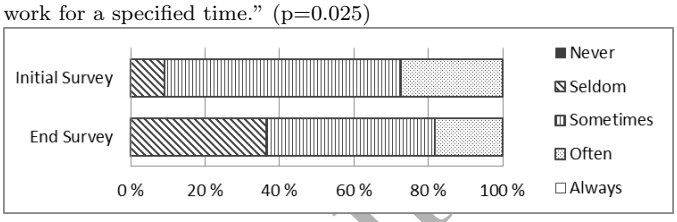

Figure 11: Case C: "It is possible to circulate tasks inside the team in order to gain knowledge." (p=0.046)

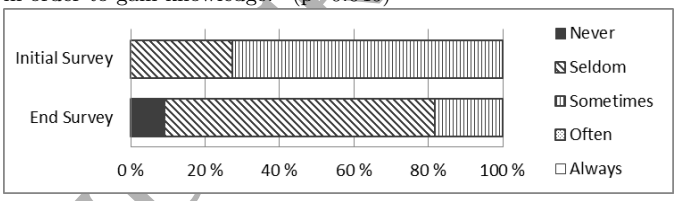

Figure 12: Case C: "The feedback received by the team is gone through systematically." (p=0.020)

Both teams were taking the first steps in making their work more agile. The new agile process was formed and it contained many new practices. The change from new practices into daily routines takes time, and it was still ongoing at the time of the end survey. Together with the quite short follow-up time, these were the factors of the negative answers. Still, in both cases, the teams decided to continue the utilization of the new working methods – despite the survey results, there were seen to be more benefits than drawbacks.

5.1.2. Case C – Refining and Spreading Agile Practices

The results of the survey comparison in case C actually present more the changes inside the whole company than only the effects of the case project. Case C also gives more answers on how to refine the current agile practices and on how they have been spread in the whole company.

Team Related Changes. As presented in Figure 10 the evaluation of the personal workload was seen to be easier than before. On the other hand, circulating tasks inside the team decreased and the feedback was inspected less systematically than before, as Figure 11 and Figure 12 illustrate.

The backlog tool was improved during the process giving a better insight to the current situation of the product and these improvements have been taken to other projects as well. Even though the backlog made work more visible, circulating tasks inside the team was experienced difficult due to different backgrounds and knowledge between the

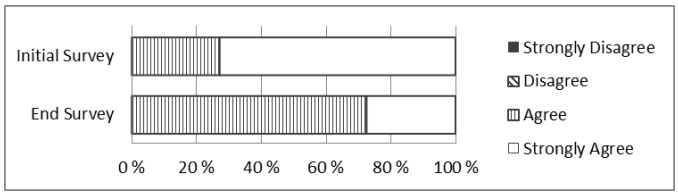

Figure 13: Case C: "Schedules and deadlines make organizing my own work and reaching goals easier." (p=0.025)

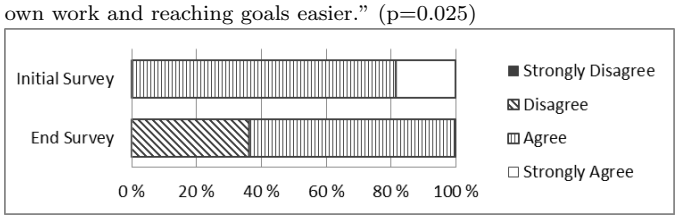

Figure 14: Case C: "The objectives of the projects are clearly divided into understandable phases." (p=0.014)

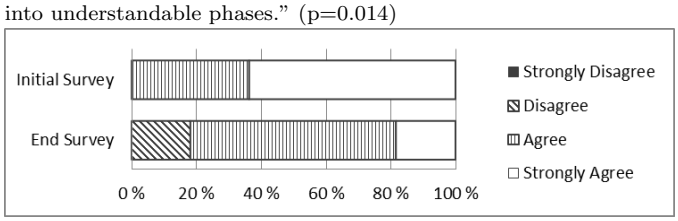

Figure 15: Case C: "The objectives can be changed in order to keep the agreed schedule." (p=0.020)

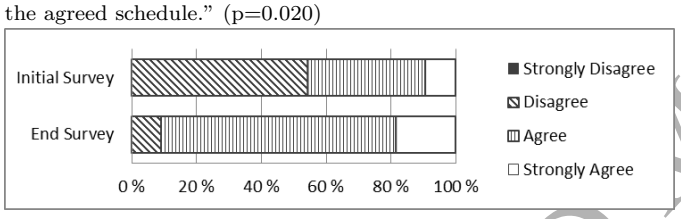

Figure 16: Case C: "The test results are documented well enough." (p=0.034)

team members. According to the end interview it was seen more probable to borrow resources from other projects than taking time to learn completely new things in order to be able to complete every task of the project. Retrospectives have not been conducted after the case project and there exists a need to concentrate more on the feedback.

According to the interviews, the case project team members would like to bring the refined practices to the whole company, but have experienced it to be more difficult than in the small case project. The improvements of the backlog tool are available to all the projects.

Changes in the Process. According to Figure 13 and Figure 14 the schedules and deadlines are making the organizing of work less easy than before and the phasing of projects is less clear than before. The possibility to change the objectives in order to keep the schedule, as presented in Figure 15, is not as good as it used to be, but the testing is better documented than before, as presented in Figure 16.

According to the interviews, the company-wide instructions for phasing and schedules of projects are currently under work, and hence they are still somewhat unclear and changing from project to project causing the process to be somewhat unclear, too. When the customer is external, the customer presence – which is seen important – is not always at as good level as during the case project. On the other hand, after the case there has been focus on testing practices and involving the testing better already in the design phase.

In case C, the pilot project was really short, but gave positive indications. After the pilot, there have been slipping from the practices, which explains the more negative results in the end survey. All in all, the answers in case C, which was somewhat familiar with agile practices even before the case project, were more positive than in the cases A and B.

5.1.3. Efficiency and Feedback - Staying the Same

Since the number of people taking part into surveys was quite small, the interpretations of the surveys had to be careful. Mostly there were no significant changes between the surveys, but a few things which did not change are highlighted here.

According to the surveys, the developers did not experience positive changes in efficiency and productivity, which are often related to agile methods. The lengths of the case projects were quite short and the only measurements for the efficiency and productivity were the opinions. On the other hand, the visibility was improved and it gave more understanding on how the project proceeded.

Feedback was one of the original issues in all the companies. The review and retrospective offered good possibilities to improve feedback. Still, it was experienced similar to what it had been. In cases A and B, the customer was not present in reviews and the implementation of changes agreed in retrospectives proved to be difficult. In case C, the retrospectives were not organized after the case project ended.

The R&D process in each company was improved and made closer to developers. The processes were still under development in the end of the case projects, making it a little bit unclear for the developers. Still, all the teams decided to continue to utilize the developed new agile working methods and develop them further.

5.2. Team Specific Experiences

In the practice survey, the team members were inquired about the difference in regard to former practices and the usage, potentiality and usefulness of each practice, but were also encouraged to give their own free comments about the practices. The new or refined practices were seen useful and potential at least to some extent. Each practice was questioned separately, and a sum of all the practices in each case is presented in Figure 17.

In case A, the experiences of the team were divided. A little over half of the team considered that the usage of

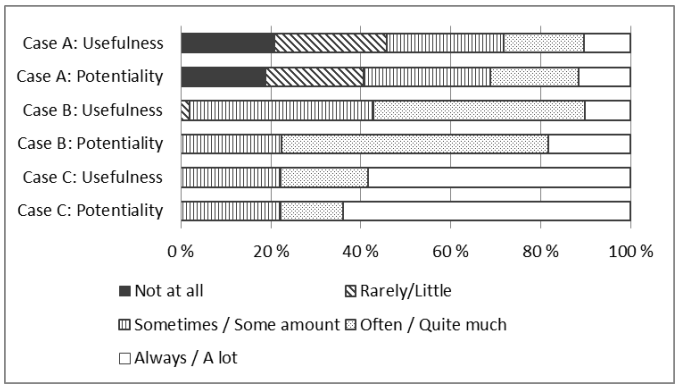

Figure 17: Usefulness and potentiality of practices in the cases.

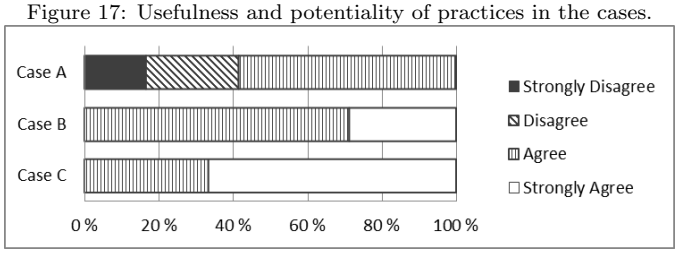

Figure 18: "Usage of the practices of the case projects would be beneficial also in the future projects".

the practices would be beneficial also in future projects, as presented in Figure 18. According to the comments in the survey, the main benefit was seen in the planning of work together and thinking of priorities. According to the interviews, the communication inside the team had also improved notably. The dropped maintenance hour was naturally seen to be the least useful practice, but also the usefulness of status meetings was questioned, since they often included information which was already familiar or not seen necessary. These two practices decrease the usefulness and potentiality in Figure 17. In the backlog tool, a graphical presentation of the progress of the product was missing, and it would have been beneficial according to the open answers. The idea of the backlog was seen beneficial, despite the lack of the tool. Some criticism was also presented to the retrospectives: they were seen useless, since the changes agreed in the retrospectives were not implemented. In the future, the team would like the process to be more release-based, where releases are more seldom than the two-week iterations.

Figure 18 presents that in case B, all of the team members agreed that the practices would be beneficial also in the future projects. According to the survey, the usage of backlogs and retrospectives were seen to be the areas of improvement in the future. The project backlog was utilized mainly by the project management, and thus the overall picture was not clear for the developers, who focused more on the implementation of the individual tasks. According to the interviews, the development process is currently better defined and the tools are similarly utilized by all the team members, whereas before the case project the working methods and tools differed between software and hardware teams hindering the collaboration. The new process has improved the visibility of the schedules and of the interdependencies between the work of developers. Both splitting the work into small enough pieces for an iteration and keeping the tasks up to date were seen difficult at the beginning of the project, but improved by the end of the project, as the benefits of common understanding became clearer and the practices more familiar. The most focal benefit was the improved transparency over the work in progress. Introducing and improving the new practices made the developers think more about the tasks, their sizes and their pacing and improving the product development process.

Also in case C, all of the team members agreed that the practices would be beneficial also in the future projects as presented in Figure 18. According to the interviews, the practices were used more efficiently than before. The process was clarified in the case project: for instance, review and retrospective templates were formed, the definition of done was defined for every requirement, and the used backlog tool was improved to better fit the needs of the team. On the other hand, the developers were also involved in other projects and thus the unknown resources posed a problem especially to the planning and self-organization of the team. The case project was quite small in terms of length and team size, and according to the interviews the challenges were more on how to spread the practices to other projects inside the company after the case project.

5.2.1. Practice Specific Experiences

The usage of *iterations* was new in cases A and B, and thus queried only in them. In these teams, the iterations were considered to give transparency over the project and the work of others. It also made the work more jointly planned, made the developers to take more responsibility and helped to organize their own work and to estimate their workload.

The planning meeting enhanced the prioritization of tasks and taking into account what other developers were working on and helped to understand their own workload. When the whole agile mindset was a new thing, i.e. in cases A and B, planning was often only going through what everyone had thought to accomplish inside the iteration and did not include the team led creation of tasks. The usefulness came then more from the possibility to align the work, instead of planning it together. *The review* of the work done in an iteration gave the developers a better view of what has been achieved, but in case A the difference from status meetings was not clear enough. The review meeting clearly had potential to improve by focusing more on the whole product instead of individual tasks.

The status meetings were considered to be a valuable time to change opinions and transfer information. The backlog tool was also used in them and it enhanced traceability. *The retrospective*, which was a new practice to all the teams, enhanced discussion about working practices, but the difficulty was the implementation of agreed

changes – it was slow and sometimes it did not work at all. Thus, the usefulness and the potentiality of this practice were among the weakest ones.

Especially when *the backlogs* were used for the first time, i.e. in cases A and B, they were considered to need more practice to get the full potential out of them. In the case A, the tool itself got a lot of criticism during the project, and in the survey it was seen less useful than the idea of a backlog. In case B, the product backlog was mainly used by only those who were managing the prioritization of features and requirements. The product backlog was not aiding the defining of the tasks enough for the planning, whereas the iteration backlog provided a good list of tasks currently under work. In the case C, the backlog tool was improved and new features were added, which made it more beneficial than before. In all the cases, the backlogs eased the project follow-up.

The maintenance hour was used only in case A and it was dropped in the beginning of the project. On the other hand, some stated, that it might have helped, if other departments would have been informed better about the new practice and everyone had agreed to truly experiment the practice.

6. Discussion

In order to answer RQ1, i.e. how to tailor agile methods in embedded system development, the two main challenges are explained and guidance for introducing agile practices into embedded system development is presented in Section 6.1. The benefits and drawbacks noted in the three cases are discussed in Section 6.2 also in the light of the known benefits and challenges of agile methods in software development. This provides an answer to RQ2.

6.1. Tailoring Agile Methods to Embedded System Development

Based on the three cases, there are two main challenges that the embedded system development poses to the agile methods: 1) the slow nature of the hardware development and 2) the specialization of the team members in the hardware or software tasks.

The slow nature of hardware development was seen in the ways the teams decided to tailor the practices. For all three teams, there was not a working product to be implemented in the end of every iteration and there was a need to organize the work in a third level on top of iteration and product backlogs. In cases A and B, the status meetings were less frequent than once a week.

The specialization could be seen in the original situation: the product development in cases A and B were divided into teams or groups based on the specialization. On the other hand, it also affected the tailoring of the planning meeting for all three teams: the tasks were defined by each discipline or individual representing the discipline already before the planning meeting, and the meeting concentrated more on the prioritization and taking into account the interdependencies between the tasks.

The case evidence from the three cases enforces the second part of the presented theory, i.e. that there are special characteristics, which need to be taken into account when utilizing agile methods in embedded system development. Similar characteristics have been already noted in the embedded software development, but the solutions have not altered the ways of working in the hardware development side and have instead focused on the software development or their tools. When also hardware development is present, these characteristics create barriers for utilizing the agile methods as specified in software engineering and require tailoring of the methods.

6.1.1. General Recommendations

Based on the three cases, recommendations were formed in order to be followed when implementing agile methods for the first time to embedded system development. Especially the four first recommendations are embedded system specific and affected by the recognized challenges of slow nature of hardware development and the specialization of team members. Since there were seen also similarities between software and embedded system development, the three latter recommendations are more general ones, but must be taken into account also when tailoring the agile methods to embedded system development.

1) Take into account the different natural cycle lengths for development. Form a consensus between the quickly changing software and slower hardware development. Take also into account how often the priorities of the requirements change and new requirements emerge when deciding the iteration length. Consider the amount of meetings and iteration length to be suitable for both the software and the hardware development.

2) Create team-driven agile practices inside the iterations. Redefine the team to consist of all the members in the project. Start with implementing simple iterative practices to get the wheel going. When it works, it is easier to add more sophisticated practices. Be still aware that you are not implementing plan-driven development inside iterations – make the plans only to the extent required in each phase.

3) Accept the different knowledge between developers and build on it. In embedded system development, different knowledge is present. Organize work planning according to disciplines, but make sure that the work of different disciplines is aligned and understood. Make it possible for everyone to understand the work of others better, for example through different meetings, in order to have more people who will be able to solve the arising problems and to be able to prioritize their own work from viewpoint of the whole product.

4) Define the progress based on work, not schedules or documentation. Even though it is not possible to present a working product at the end of each iteration, review the work done. The aim is to present e.g. simulations or other work products but, even if this is not possible, at least visualize the progress of the product. Take care of longer-term goals such as releases, which might be less frequent than the iterations, but need to be managed e.g. in the backlogs.

5) Clarify the reasoning behind the utilized practices together in the team. The key to a successful introduction of new practices is to understand why the current practices are altered. Transparency, for example, will help everyone: the project manager will see the real status of the project, whereas the developer will be able to organize his work according to the needs of other developers thus minimizing the waiting time and frustration. Also the visibility of external disturbances, such as manufacturing, component delivery waiting time or new requirements, will increase, which provides further understanding about the bottlenecks of the project.

6) Try more advanced agile techniques. When the team is familiar with self-improvement, it is more eager to try out new solutions. In embedded system development e.g. platform-based design will offer quicker solutions for testing new features and test-driven development can improve the development process. The team can try new techniques and select the ones that work best for them.

7) Involve the whole organization. In order to truly understand what the product is, input is also required from other members of the organization than just the developer team. If the end customer is not clear, e.g. business and marketing people will be able to shed light on where the requirements have originated from. When the process is changed to utilize agile practices, it will affect also the ways the developers co-operate with other departments, and without support from the whole organization some practices may be difficult to implement effectively.

These guidelines formed by the experiences of the three cases are considered to help other teams in the embedded system domain to conquer the challenges detected in this work and acquire the benefits of combined agile and embedded system development.

6.2. Effects of Agile Methods in Embedded System Development

In software development, the main benefits of agile methods are seen to be better productivity, possibility to manage changing priorities, and improved team morale and communication [12] [13] [14].

Even though similar effects would be welcome in embedded system development, all of them were not apparent. The productivity was not measured, but the none of the three teams experienced enhancements in the productivity. Especially for cases A and B where agile mindset was a new thing, this could be partly due to the short follow-up time, upon which the new practices were only introduced and not thoroughly adopted as routines. Learning new ways of working takes time, and can diminish the productivity for a while.

In cases A and B, the interdependencies were taken into account better between the different disciplines. Not only the changes of priorities were now managed, but also the priorities were reconsidered from the whole product point of view. Also teamwork and communication enhanced, and the most focal benefit from that was the system-wide understanding. This affected also the diminished need of documentation, similar result that has been seen in software development at least in a survey conducted in Ericsson [31].

In case C, which focused in refining and spreading agile practices to the company, the most focal benefit was seen to be more efficient usage of the practices in the case project. On the other hand, similarly to software development [14], there were challenges in scaling.

In cases A and B difficulties of implementing the changes agreed in retrospectives can tell about resistance to change, which is one of the challenges noted in software engineering [12] and in case B also difficulties with other departments were note, similarly to software engineering [13]. An embedded system specific challenge was encountered when trying to split the work to small enough tasks to be accomplished in one iteration. The slow nature of hardware development created this challenge, and as a solution, it was quite common that some tasks were not accomplished within one iteration. This steered the focus away from the working product – as specified in agile values – into work accomplished.

According to the experiences of the three case studies, one should not expect dramatical improvement in productivity when taking the agile methods into use in embedded system development. The benefits come from the improved system-wide understanding, which enables better prioritization of the tasks. This is enabled by the improved teamwork and communication.

The analysis shows that the findings of our case study are mostly aligned with the previous research on the benefits and drawbacks of the agile methods in software development. However, there are also some abbreviations e.g. no notable increase in productivity, which may be partly due to the short observation time within which the practices were still forming and not adopted as routines. The similar the findings strengthen the theory set in the beginning of this paper saying "agile methods can offer benefits to embedded system development similar to benefits in software development". The found abbreviations suggest that also the second part of the theory considering the special nature of embedded system development might also be true.

7. Conclusions

In this paper, a case study of three cases on the adaptation of agile practices into embedded system development was presented. All of the three case projects included hardware development and one of them consisted of only hardware development. Thus the case study offered a possibility to expand agile development practices outside software development, where it originates from.

The case study consisted of three parts: 1) understanding the process in the companies before introducing or refining agile practices through interviews and a survey, 2) observing and assisting in the method development and 3) understanding the changes and experiences of the case project through interviews and a similar survey to the initial one.

The case evidence strengthened the theory presented in this work: *agile methods offer benefits to embedded system development similar to benefits of software benefits, but the methods need to be tailored due to the special characteristics of embedded system development.* From the special characteristics, there were especially two recognized: 1) the slower nature of hardware development, which was seen e.g. in the absence of working product in the end of every iteration and 2) the different knowledge between developers, which e.g. required tailoring the planning meeting according to the disciplines.

Based on the case company experiences, recommendations were formed for the tailoring of the agile practices into embedded system development, when also hardware development is included in the scope. Forming a unified project team of different disciplines provides better understanding over the work of others. Even though circulating tasks is not possible due to different disciplines, planning the work together facilitates the prioritization of tasks and makes it possible to take into account the interdependencies between different disciplines. Since it is often impossible to present a new version of working product in small intervals, the work done is the best available measure of progress. The product is presented in the end of each iteration, preferably in the form of demonstrations or simulations, or at least through closing the tasks and requirements together. The naturally longer development cycle of hardware development must be taken into account also in e.g. iteration length or meeting frequency. After the first iterative practices are in place, even more advanced agile techniques, such as platform-based design, can be implemented. In order to successfully adopt agile practices, also other than embedded system development related recommendations must be followed: the motivation for the change must be found and the whole organization should be involved.

Compared to the typical experienced benefits of agile methods in software development, there is a change of emphasis. According to the experiences of the three cases, productivity and efficiency was not improved as has been experienced in software development. This may have been affected also by the short length of the observation time. Similarly to software development, teamwork and communication were enhanced, and in the long run the enhanced teamwork and communication will probably lead to better productivity and efficiency. In short term, the improved teamwork and communication lead to better system-wide understanding and diminished need for documentation, and enabled taking interdependencies better into account when defining and changing priorities of the tasks.

Dividing the work into iterations especially on the hardware side was experienced difficult and all the tasks were not completed in one iteration. Utilization of agile methods in embedded system development challenged most the agile value of the working product: the focus shifted from the documentation to the work accomplished, instead of the working product.

In the case studies it was noted, that the agile value of respecting individuals and interactions is a key for successful adoption of agile practices. The team – and each individual in it – is responsible for the development of their own working methods. Obtaining the full benefits of agile methods in embedded system development takes time but it is worth the effort.

Acknowledgment

The research reported in this article has been conducted as a part of AgiES (Agile and Lean Product Development Methods for Embedded ICT Systems) project. The project is carried out in collaboration with Finnish Institute of Occupational Health and industry partners BA Group, FiSMA, Lindorff Finland, LM Ericsson, Neoxen Systems, Nextfour Group and Nordic ID. The project is mainly funded by Tekes – the Finnish Funding Agency for Technology and Innovation.

References

[1] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, Agile manifesto, in: http://agilemanifesto.org/, 2001.

[2] M. Kaisti, V. Rantala, T. Mujunen, S. Hyrynsalmi, K. Könnölä, T. Mäkilä, T. Lehtonen, Agile methods for embedded systems development – a literature review and a mapping study, EURASIP Journal on Embedded Systems 2013 (1). doi:10.1186/1687-3963-2013-15.
URL http://dx.doi.org/10.1186/1687-3963-2013-15

[3] W. Royce, Managing the development of large software systems: Concepts and techniques, in: IEEE WESTCON, 1970.

[4] C. Larman, V. Basili, Iterative and Incremental Development: A Brief History, Computer 36 (6) (2003) 47–56. doi:10.1109/MC.2003.1204375.

[5] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice Hall, 2001.

[6] K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley Professional, 1999.

[7] C. Larman, Agile & Iterative Development: A Manager's Guide, Addison-Wesley, 2003.

[8] T. Dingsøyr, S. Nerur, V. Balijepally, N. B. Moe, A decade of agile methodologies: Towards explaining agile software development, Journal of Systems and Software 85 (6) (2012) 1213 – 1221, special Issue: Agile Development. doi:http://dx.doi.org/10.1016/j.jss.2012.02.033.
URL http://www.sciencedirect.com/science/article/pii/S0164121212000532

[9] A. Cockburn, Agile Software Development, 2nd Edition, Addison-Wesley, 2007.

[10] P. Kettunen, Adopting key lessons from agile manufacturing to agile software product developmenta comparative study, Technovation 29 (67) (2009) 408 – 422. doi:http://dx.doi.org/10.1016/j.technovation.2008.10.003.
URL http://www.sciencedirect.com/science/article/pii/S0166497208001302

[11] M. Laanti, Agile and wellbeing – stress, empowerment, and performance in scrum and kanban teams, in: System Sciences (HICSS), 2013 46th Hawaii International Conference on, 2013, pp. 4761–4770. doi:10.1109/HICSS.2013.74.

[12] VersionOne. The 10th annual state of agile development survey [online] (2015).

[13] P. Rodríguez, J. Markkula, M. Oivo, K. Turula, Survey on agile and lean usage in finnish software industry, in: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '12, ACM, New York, NY, USA, 2012, pp. 139–148. doi:10.1145/2372251.2372275.
URL http://doi.acm.org/10.1145/2372251.2372275

[14] A. Begel, N. Nagappan, Usage and perceptions of agile software development in an industrial context: An exploratory study, in: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, ESEM '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 255–264. doi:10.1109/ESEM.2007.85.
URL http://dx.doi.org/10.1109/ESEM.2007.85

[15] B. Greene, Agile methods applied to embedded firmware development, in: Agile Development Conference, 2004, 2004, pp. 71–77. doi:10.1109/ADEVC.2004.3.

[16] N. Van Schooenderwoert, Embedded agile project by the numbers with newbies, in: Agile Conference, 2006, 2006, pp. 13 pp.–366. doi:10.1109/AGILE.2006.24.

[17] M. Fletcher, W. Bereza, M. Karlesky, G. Williams, Evolving into embedded develop, AGILE Conference 0 (2007) 150–155. doi:http://doi.ieeecomputersociety.org/10.1109/AGILE.2007.25.

[18] U. Eklund, J. Bosch, Applying agile development in mass-produced embedded systems, in: C. Wohlin (Ed.), Agile Processes in Software Engineering and Extreme Programming, Vol. 111 of Lecture Notes in Business Information Processing, Springer Berlin Heidelberg, 2012, pp. 31–46. doi:10.1007/978-3-642-30350-0_3.
URL http://dx.doi.org/10.1007/978-3-642-30350-0_3

[19] M. Kaisti, T. Mujunen, T. Mäkilä, V. Rantala, T. Lehtonen, Agile principles in the embedded system development, in: 15th International Conference on Agile Software Development XP2014, 2014.

[20] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still, The impact of agile practices on communication in software development, Empirical Software Engineering 13 (3) (2008) 303–337. doi:10.1007/s10664-008-9065-9.
URL http://dx.doi.org/10.1007/s10664-008-9065-9

[21] L. Cao, How extreme does extreme programming have to be? adapting xp practices to large-scale projects, in: 37th Hawaii International Conference on System Sciences, 2004, pp. 1–10.

[22] J. Ronkainen, P. Abrahamsson, Software development under stringent hardware constraints: Do agile methods have a chance?, in: 4th International Conference on Extreme Programming and Agile Processes in Software Engineering, 2003, pp. 73–79.

[23] U. Eklund, H. Holmström Olsson, N. J. Ström, Industrial challenges of scaling agile in mass-produced embedded systems, in: T. Dingsøyr, N. Moe, R. Tonelli, S. Counsell, C. Gencel, K. Petersen (Eds.), Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation, Vol. 199 of Lecture Notes in Business Information Processing, Springer International Publishing, 2014, pp. 30–42. doi:10.1007/978-3-319-14358-3_4.
URL http://dx.doi.org/10.1007/978-3-319-14358-3_4

[24] R. K. Yin, Case Study Research: Design and Methods, SAGE Publications, 2003.

[25] P. Runeson, M. Höst, A. Rainer, B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples, Wiley Blackwell, 2012.

[26] B. Kitchenham, L. Pickard, S. L. Pfleeger, Case studies for method and tool evaluation, Software, IEEE 12 (4) (1995) 52–62. doi:10.1109/52.391832.

[27] C. B. Seaman, Qualitative methods in empirical studies of software engineering, IEEE Trans. Softw. Eng. 25 (4) (1999) 557–572. doi:10.1109/32.799955.
URL http://dx.doi.org/10.1109/32.799955

[28] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2) (2009) 131–164. doi:10.1007/s10664-008-9102-8.
URL http://dx.doi.org/10.1007/s10664-008-9102-8

[29] P. Brereton, B. Kitchenham, D. Budgen, Z. Li, Using a protocol template for case study planning, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, British Computer Society, Swinton, UK, UK, 2008, pp. 41–48.
URL http://dl.acm.org/citation.cfm?id=2227115.2227120

[30] B. Curtis, H. Krasner, N. Iscoe, A field study of the software design process for large systems, Commun. ACM 31 (11) (1988) 1268–1287. doi:10.1145/50087.50089.
URL http://doi.acm.org/10.1145/50087.50089

[31] K. Petersen, C. Wohlin, The effect of moving from a plan-driven to an incremental software development approach with agile practices, Empirical Softw. Engg. 15 (6) (2010) 654–693. doi:10.1007/s10664-010-9136-6.
URL http://dx.doi.org/10.1007/s10664-010-9136-6

[32] K. R. Neil J. Salkind (Ed.), Encyclopedia of Measurement and Statistics, 0th Edition, Sage Publications, Inc., 2007. doi:http://dx.doi.org/10.4135/9781412952644.
URL http://dx.doi.org/10.4135/9781412952644

Biography

Kaisa Könnölä has M.Sc. (Tech.) degree, majoring in telecommunication, from University of Turku in 2004. She has several years' experience as software professional in industry mainly from software integration. She has been working in Technology Research Center at the University of Turku since 2013 as a researcher. Her research interests include developing and investigating agile and lean methods in embedded system and space system development.

Samuli Suomi received the B.Sc. (Tech.) degree in computer science and the M.Sc. (Tech.) degree in software engineering from the University of Turku, Turku, Finland, in 2013 and 2015, respectively. His work as a project researcher in Technology Research Center at the University of Turku has focused in agile development and agile tools.

Tuomas Mäkilä works as a Senior Research Fellow at the University of Turku, Technology Research Center. He holds a D.Sc. (Tech.) degree in software engineering and wrote his doctoral thesis on the modelling techniques of software development processes. He has experience on wide variety of software engineering topics, including software development methods and tools, software architectures, and web and game development.

Tero Jokela received the D.Sc. (Tech.) degree in telecommunication from the University of Turku in 2010. His research interests include multiple antenna communications, error control coding, receiver algorithms, cognitive radio and wireless broadcasting.

Ville Rantala received the D.Sc. (Tech.) degree in electronics from the University of Turku, Turku, Finland, in 2012. Dr. Rantala has expertise in both circuit and system level digital IC design using modern EDA tools and languages such as VHDL and SystemC. His research interests include iterative development processes for safety-critical embedded system industry.

Teijo Lehtonen works as a Senior Research Fellow at the University of Turku, Technology Research Center, where he leads the Embedded and Mixed-Reality Systems research group. Lehtonen holds a D.Sc. (Tech.) degree in electronics from the University of Turku in 2009. Dr. Lehtonen has been the principal investigator in numerous research and development projects with the total value of several million euros. His research interests includes adoption of agile methods to embedded system development, and mixed-reality technologies and applications.