Könnölä *et al. EURASIP Journal on Embedded Systems* (2017) 2017:3
DOI 10.1186/s13639-016-0040-z

EURASIP Journal on Embedded Systems

RESEARCH Open Access

Can embedded space system development benefit from agile practices?

Kaisa Könnölä*, Samuli Suomi, Tuomas Mäkilä, Ville Rantala and Teijo Lehtonen

Abstract

We study in this work piloting agile practices in embedded space system development projects. The case involves three companies acting as last or next to last subcontractors in a space project.

Initial interviews and a subsequent survey revealed challenges that embedded space system development poses to the agile software development. These include high specialization and emphasis on individual performance, formal customer interface requiring extensive documentation, and the management of several simultaneous projects.

Iterative way of working is characteristic to agile systems development, but novel to the examined project teams. We observed that it enhanced team collaboration through planning and reviewing the work together and in transferring emphasis from the individuals to the team. Resource allocation between projects was taken properly into account when planning the iteration, or when cancelling an iteration for the project. Furthermore, the customer interface was tackled better by utilizing backlogs.

According to the end survey and interviews, the main benefits of the agile practices were better communication and knowledge sharing inside the team, enhanced teamwork, and setting the pace for the sometimes slowly proceeding embedded space system development. Also documentation, while it barely changed, was seen more adequate in two of the cases. Overall, the teams felt that they were given a better possibility to affect their ways of working.

The case study results show that agile practices can be applied to embedded space system development with notable benefits.

Keywords: Space system, Embedded system, Agile, Agile method, Case study

1 Introduction

Agile methods are widely utilized in software development, where these methods are known to have several benefits, such as improved efficiency and productivity [1] and improved experienced productivity and visibility through better communication [2, 3]. These methods have common background in the agile manifesto [4], formed in 2001, which defines four values: individuals and interactions, working software, customer collaboration, and responding to change. Methods such as Scrum [5] and eXtreme Programming (XP) [6], offer practices for implementation of the four values and 12 principles.

Embedded space system development has several characteristics where it differs from software development. It consists of products that have both hardware and software, which are typically unique. Typically, and especially in the European Space Agency (ESA)-initiated projects, the development involves several subcontractors, which are located even in different countries. Each subcontractor is responsible for developing a small portion of the whole product, and there are quite often small and medium-sized enterprises (SMEs) in the end of the subcontractor chain. Another typical feature of the embedded space system development is the safety-criticality, and thus, the development is limited by several standards and regulations.

In this paper, a case study carried out in three companies in embedded space system development is presented. The main topics examined were (1) the challenges the embedded space system development creates to agile methods, (2) how agile practices could be applied to embedded space system development, and (3) what are the benefits gained. In the paper, first, the embedded space system

*Correspondence: kaisa.konnola@utu.fi
[1] Technology Research Center, University of Turku, 20014 Turun yliopisto, Helsinki Finland

development is described as well as agile development. Also, the special characteristics the embedded space system development has in contrast to agile development are considered in light of previous research. Next, the case study method utilized is explained in detail followed by the results of the case study. The results give insight on the special challenges of embedded space system development, how these challenges are solved utilizing agile practices, and how the new practices were experienced. Before conclusions, the results are further elaborated and generalized.

2 Background

2.1 Embedded space system development

Embedded systems, which consist of both hardware and software, are present everywhere. Embedded systems are designed to fulfill a special task in the environment they operate. They vary from mass-produced consumer products to highly customized professional products. Embedded space systems are in the end of highly customized products: they are typically unique, and in the end only, one (or a few) product is manufactured.

The space system development consists of various fields, where *embedded space system development* differentiates from, e.g., algorithm and software development and includes development of the various instrument development for the space mission. The space as an environment brings its own challenges to the embedded space system development: the systems in space must be tolerant to heat and radiation, and the size and weight as well as the weight distribution are strictly defined, for example. This brings vast requirements to the development of embedded space systems and several new disciplines compared to more ordinary embedded system development. The systems must also be reliable, since repairing is often impossible.

In Europe, a space system development project is often originated from the European Space Agency (ESA [7]). Since ESA is funded by several countries, the implementation is also divided to the companies residing in several countries. Also, the chain of subcontractors may be long. In the end of the subcontractor chain, there are often SMEs, which are implementing only one small portion of the whole product.

In order to ensure that all the stakeholders of the project act similarly and in order to take care of the safety-criticality of the space product development, standards and regulations are utilized. The typically utilized standards in Europe are the ECSS standards (European Cooperation for Space Standardization), which include standards for project management, project assurance, engineering, and sustainability [8].

The embedded space system development is typically divided into phases defined in the ECSS standards. The phases are sequential in so called mission projects. In the beginning, the customer and the top level supplier elaborate the functional and technical requirements for the system. In the next phases the product is developed, first the engineering model and then the flight model [9]. Lower level suppliers are usually involved in the product development through a tendering process: in the invitation to tender (ITT), the requirements are already defined at a quite detailed level. The design is usually defined already, even though in high level, since the tender needs to answer to the ITT and the pricing is highly dependent on the selected components in the design.

2.2 Agile development

The most prominent definition of agile development is the one provided in the Manifesto for Agile Software Development [4]. The manifesto was formed by a group of software development professionals and proponents of lighter software development methods in 2001. The authors of the manifesto agreed upon four values and 12 principles to form a common definition for lightweight iterative ways to build software in contrast to more plan-driven approaches in the history [10]. The four values are listed in Table 1.

Iterative and incremental development (IID) where projects are divided into several, usually short and fixed length iterations, is a key concept in agile methods. The system is developed by doing planning, design, implementation, and testing work in each iteration rather than planning early and testing late in the project. New features are added to the software incrementally, and the software is always kept functional. While agile methods are a well-known example of IID, the history of IID methods starts from as early as the 1930s. Also, IID has not been limited to software development in the past: for instance, it was used in the X-15 hypersonic jet project in the 1950s [11].

While there are several agile methods, Scrum and XP are the most well-known ones [12]. All agile methods provide practical ways to achieve the goals behind the manifesto. For instance, in Scrum, the focus is in the management of work and frequent ceremonies, giving room for various ways of doing the actual development work [5].

Table 1 Agile values presented in the Agile Manifesto [4]

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

On the other hand, XP provides a collection of detailed best practices of software development [6].

According to a systematic review on empirical studies of agile software development [1], benefits were reported in customer collaboration, work processes for handling defects, learning in pair programming, thinking ahead for management, focusing on current work for engineers, and estimation. In three of four comparative studies also, productivity in terms of LOC/h (lines of code) was increased. It was also found in non-comparative studies that the subjects believed that the productivity has increased with agile methods [1].

Agile methods can have positive effects on both productivity and well-being at work, by aiming for a sustainable pace, which itself leads to a less stressful working environment [13]. According to the 10th Annual Agile Survey by VersionOne, a company developing agile lifecycle management software, the top four benefits were: (1) ability to manage changing priorities, (2) increased team productivity, (3) improved project visibility, and (4) increased team morale/motivation [3]. In a survey made in the Finnish software industry, the top three benefits were (1) improved team communication, (2) enhanced ability to adapt to changes, and (3) increased productivity [2].

Utilizing agile methods in embedded space system development has several positive opportunities worth studying. While some benefits of agile development, such as decreasing time-to-market, are difficult to take advantage of in a single subcontracted team, other benefits are arguable. For example, the ability to manage changing priorities and improved teamwork are worth pursuing.

2.3 Related work

There exists some research and experience of agile or iterative methods in space system development. In order to understand the possibilities, some of the known characteristics of embedded space system development should be also taken into account. Embedded space system development is influenced by standards and regulation due to the safety-criticality of the systems. The requirements are typically fixed already in the beginning of the project. Also the simultaneous development of hardware and software creates challenges to the utilization of agile methods.

2.3.1 Space system development

ESA has already been developing so-called concurrent engineering activities, which share some similar components to agile development, such as all the developers of different disciplines working in the same space and iterative and incremental development. Still, the concurrent engineering activities are utilized only in the early phases of a project, i.e., before the implementation by the subcontractors [14].

In [15], agile methods and their applicability to space system engineering are considered. Some agile practices and methods, such as test-driven development (TDD), Scrum, and XP are seen to be very useful and should be given consideration. The authors also point out that hybrid approaches, where agile practices are combined with existing formal methodologies and engineering standards, should also be practiced, e.g. in such a way that agile practices are utilized for areas requiring knowledge discovery whereas formal methodologies are utilized in known and repeatable processes. It is stated that more supporting data needs to be collected with respect to the specific applicability of agile practices in the space industry, e.g. through case studies, mapping formal processes with agile methods, and establishing a database of lessons learned [15].

In [16], Scrum was utilized in the development of ground segment software. When the contract was based on a firm and fixed price, it was seen that mapping between requirements and user stories was cumbersome and took more time in the planning phase. On the other hand, significant benefits were seen in the higher quality, earlier detection of possible problems, and the early visibility to the results of the project.

Through an industrial case study, Ahmad et al. conclude that there are no strict contradictions between agile methodologies and the development of software for ECSS-regulated systems. They state that the ECSS standards agree with agile thinking in certain areas and that many agile practices, such as frequent review and planning meetings and TDD, are beneficial in the space software projects [17].

2.3.2 Standards, regulations, safety-criticality, and fixed requirements

Space projects are usually heavily regulated. For instance, in ESA projects, the ECSS standards define milestones, require mandatory documentation, and set limitations for the component selection. Because the projects have many subcontractors, fixed and detailed requirements are formed in the beginning of the projects. Requirements might be changed, but there are slow and formalized processes in place for doing so.

In [18], it is stated that agile is highly suitable even in safety-critical software development, as long as the practices are tailored to meet the needs of the regulated environment and supported with appropriate tools. The found benefits of agile development in regulated environments included enhanced quality, mitigated risk, and end-to-end traceability.

Sidky and Arthur present a three-stage process for identifying the agile practices that an organization can most benefit from while maintaining the compliance of mission and life-critical requirements. The authors state minimal

documentation and evolutionary requirements as examples of agile practices that are not suitable for mission and life-critical systems [19].

Agile development can be beneficial in projects even if the requirements are rigid. Hoda et al. state in their case study that, even though there was doubt about the suitability of agile methods in projects without frequent requirement changes, participants still valued other agile practices. One team that combined Scrum and XP in a requirement-stable environment found still use for planning sessions, release planning, story boards and retrospectives. For instance, in planning sessions the original requirements were transformed into stories and the stories from lower priority requirements were postponed to later iterations [20].

2.3.3 Embedded system development

There are successful experiences of agile development of embedded systems, although it is common in many cases to concentrate only on the software side of the development process. In the literature review [21], there were 18 case studies or experience reports. From these, six utilized XP as their main method, three utilized both Scrum and XP, two utilized Scrum, one utilized the practice of test-driven development, and one utilized hardware-software co-design. General agile thoughts were reported as a basis in three of the cases, and one utilized agile requirements engineering. In one of the articles, a new method called document-driven development was created. In many of these cases tailoring [22–24] or adaptation [25, 26] of agile methods was seen essential, and some decided to utilize only some specific agile practices [27, 28]. Also it was noted that agile methods should not be followed dogmatically [29, 30] or at least the practices should be carefully selected [31] or the methods should be concentrated on the embedded domain-specific requirements [21, 32].

When both hardware and software development are wanted to be included in the same method utilization, challenges such as interdependencies between software and hardware tasks, specialized team members, and long development cycles of hardware development need to be addressed. The core ideas of agile development can be maintained in embedded system development, e.g., by building frequent prototypes of hardware, investing in modular designs and using simulations [33].

According to previous research, several agile methods and practices (e.g. XP, Scrum, TDD) look promising from the embedded space system development's point of view. Even though all practices, such as minimal documentation, are not possible in a safety-critical environment, they can even enhance the quality. Also, in a stable requirement environment, the utilization was seen useful. In space system development and in embedded system development, there is a need to carefully select or tailor the practices. This was also noted in our recent study [38]. A case study can enhance the knowledge over the challenges and benefits of agile methods in embedded space system development.

3 Methodology

Agile methods are popular in software engineering, and the benefits gained there could be beneficial also in other industries. When applying a method to another domain, the special characteristics of the new domain must be taken into account. The hypothesis is that agile methods can give similar benefits as in software industry to the embedded space industry and also help to solve the special challenges of the embedded space system development.

The research questions are:

> RQ1: What are the challenges the embedded space system development poses to agile development especially in the case companies?
> RQ2: How the agile practices can be used and tailored in order to conquer the challenges in the space companies?
> RQ3: What are the benefits and drawbacks of agile practices in everyday work experienced by the teams in the case companies?

3.1 Case study design

A case study was selected as an appropriate way to research how agile methods could be applied to the embedded space system development. In general, a case study is an empirical inquiry aimed at investigating contemporary phenomena in their real-life context [34]. Following a case study protocol helps to ensure the quality of a case study, including taking into account the theoretical basis, using triangulation, presenting the chain of evidence with traceable reasons and arguments, and fully documenting and formally reporting the case study [34]. Engineering work in different disciplines reminds each other, and as software development is the origin of agile methods, recommendations of case studies in software engineering were utilized, especially the ones defined in [35].

This case study is a multiple case study of three individual cases in three companies, where in each company, a project team or several project teams were the units of analysis. The case study is clearly seeking to improve the current development process.

3.2 Data collection and analysis

The case study was designed to comprise of three parts, which can be quite directly related to the research questions: (1) understanding the special characteristics of the embedded space system development from the agile practices applicability point of view, (2) tailoring the agile

practices into the embedded space system development, and (3) understanding the value of the agile practices in the embedded space system development. The collection of data used multiple sources of data: existing documentation, surveys, observations, and several interviews.

3.2.1 Understanding the embedded space development challenges

The first phase of the case study consisted of interviewing and conducting a survey in three different organizations working on the embedded space systems industry. RUAG Space Finland was the largest company, with about 30 employees, of the three companies taking part into this research and its key product areas are spacecraft control electronics, electrical power subsystems, drive electronic units, and related test equipment. The other two smaller companies, which have under 10 employees, are focused not only to the space system development but also to other industrial development: Aboa Space Research designs and manufactures electronic systems for various measurement and data handling purposes as a part of clients R&D activities for space and industrial applications, whereas Harp Technologies develops RF, microwave, and millimeter wave technology including also design services in embedded electronics and mechanical design. The interviews in these three companies were based on interview guides and included questions about the working practices, such as typical projects, requirements, and specifications, teamwork, current processes, documentation, communication, and the bidding process. The interviewees were from different positions varying from CEO or general manager through lead developers to developers as presented in Table 2. The survey was conducted to the whole personnel of the company, and the number of respondents are presented also in Table 2. The survey consisted of similar themes to the interview guides.

The interviews were conducted by two researchers, where one researcher was taking notes, which were complemented with recordings where necessary. This interview data was analyzed by differentiating the comments which were seen either as challenges in the current ways of working or challenges for agile methods. The survey focused more on the working methods and agile values and gave insight of the current situation in the company. The statements with most negative answers were collected and discussed with company representatives. The results of the first phase were collected to an initial state report, which was utilized as a basis for the new working methods.

3.2.2 Tailoring the agile practices in pilot projects

The second part of the research focused on developing new ways of working in the selected pilot projects in each of the companies. In space system development the projects tend to last quite a long time, and therefore of the main criteria in selecting the pilot project was the availability of the projects during the research time. In RUAG Space Finland, later referred as case A, the pilot project was selected to be a small project with perhaps less strict requirements, in order to be able to more freely test and tailor the development process. After a while, also another project in RUAG Space Finland decided to start utilizing the same practices, partly due to half of the developers being the same and partly due to the experienced usefulness of the methods. In Aboa Space Research, case B, the pilot project was selected to be an ongoing project. In Harp Technologies, case C, the pilot project was selected to be such, that it involved several developers, whereas some projects in this company were implemented by only a few developers. The case data for each case is presented in Table 3.

According to agile values, individuals and interactions were valued and the teams were able to affect the working methods throughout the project. Especially in the beginning, the researchers provided a basis for the new process, but the team defined the details. Later, the researchers focused on observing the meetings of the new working method. Even though the researchers provided ideas for the solutions to the challenges, the team was responsible for developing the process further. In the end, all the teams had a new process, which was seen as a basis for the new projects to come.

3.2.3 Understanding the effects of the new agile practices

The last part of the research was essential to understanding the real effects of the new practices. First, a survey similar to the initial one was conducted, with some additional questions about the practices utilized. The responses of the initial and end survey were compared in order to find the differences between them. Then, interviews were conducted in order to understand what were the reasons

Table 2 Initial interviews and surveys in companies

	RUAG space Finland	Aboa space research	Harp technologies
Interviewed roles	General manager, project manager, product assurance manager, system designer, and designer	CEO, part-time, and full-time developers	CEO and developer
Number of interviewees	8	7	4
Survey respondents	26/30 (87 %)	6/9 (67 %)	5/6 (83 %)

Table 3 Pilot project data

	Case A (RUAG space Finland)	Case B (Aboa space research)	Case C (Harp technologies)
Pilot project(s) observation length	09/2014–01/2015, 11/2014–05/2015	10/2014–04/2015	05/2015–09/2015
Pilot project team size	5, 6	6	4

for the changes and how the practices were experienced during the project.

4 Results of the case studies

The results of the case studies can be divided into three parts. The first part focuses on the challenges and opportunities, which were found from the initial interviews and the initial survey. The second part consists of agile practices utilized for answering the found challenges. The third part of the results is about the experiences of the teams about the new agile practices based on the end survey and the end interviews.

4.1 Challenges and opportunities to agile practices in embedded space system development

In order to understand how the domain of embedded space system development affects the ways of working, a survey and interviews were carried out. The ways of working, which contradicted with the agile values or iterative and incremental development were grouped to different categories, which are presented in Table 4 as well as with which characteristic of agile development they contradict with.

4.1.1 Challenges inside the teams

In space system development, there are several areas where specialist knowledge is required. The requirement for different specialists comes from the vast requirements of the space systems due to the environment where they must function: their development requires understanding the effects of heat and radiation, for example. Also, already in the bidding phase, the tender must typically include named persons for each special area. Since all the developers do not have same background and understanding over the special areas, the knowledge difference between the developers, i.e., *the specialization*, can hinder the interaction between the team members. Thus, the specialization often leads to the situation where the work is conducted individually and not as a team. It can also lead to a situation where knowledge is not shared, since it is not seen beneficial.

In Europe, especially in smaller countries, the space system development is often based on the ESA projects. The duration of space system development projects is often several years long and has multiple subcontractors in several countries. It is typical that there are times during the implementation where the part of one subcontractor does not proceed, since it is not in the critical path. Also, when the specialization of the developers is taken into account, there are simultaneously several projects active for each developer, and all the customers expect that their project is still proceeding. This has led to a situation where a developer implements *multiple projects simultaneously*. This often leads to a situation where the developer does not have time to take part in the meetings of all the projects and thus is not aware of the progress of each project, which endangers the work as a team.

The development is divided into milestones, which tend to be several months long. This schedule is derived from the customer and originate from the ECSS standards. The length of the milestones are several months long, and there seems to be a hurry in the end of the milestones. On the other hand, the length of the milestones leads often to a situation where the developer designs his share of the product for a long time without receiving feedback. This

Table 4 Embedded space development challenges versus agile values

	Iterative and incremental development	Individuals and interactions	Working software (product)	Customer collaboration	Responding to change
Specialization of team members		x			
Multiple simultaneous projects		x			
Emphasis on individual performance		x			
Formal customer interface				x	
Required customer documentation			x		
Detailed upfront planning	x		x		x
The difficulty of making changes					x
Schedule bends instead of requirements					x
Different tools and tool ownership	x			x	

has put *emphasis on individual performance* instead of the team and can hinder the interaction within the team.

The team-related challenges rise from the space system development: the main reasons for this are the standards and the ways of working in the distributed settings.

4.1.2 Challenges in the process

Due to the subcontractor chain in ESA projects, the customer in the projects is typically a larger company in the space industry. Another typical possibility, that the space companies had was a university that has ordered, e.g., a feasibility study for their projects. The discussion especially with the larger companies is conducted often through the project managers. It is common that the customer is even in a different country and the discussion is through e-mails, documentation, and formal meetings. The emphasis is more on the process than in the customer collaboration—for example, it takes time to get the official changes to the requirements even though they are agreed in a meeting. Then the subcontractor is not certain whether they can proceed with the approach agreed in the meeting or should wait for the official documentation. Another typical issue to be dealt with is the nonconformance process, which takes quite a long time. The *formal customer interface* challenges the value of customer collaboration.

The ECSS standards define in a considerably detailed level the required documentation in the projects. Even though the documentation can be seen as a view for the product before any prototype exists, significant time is spent finalizing the documentation during and after the implementation. *The required customer documentation* seems to fulfill mostly the documentation needs, but it can shift the focus of the work from the product to the documentation.

Already in the tender, *upfront planning* is required, since the tender includes the budget of the project, and to this, the selected components have a high impact. Also, the ECSS standards require that the specifications are designed in quite detailed level before the implementation. Due to the small selection of space-approved components, their delivery time can be long, and thus, they need to be ordered in the early phases of the design. This makes easily the implementation plan-driven instead of iterative and incremental.

In embedded space system development, it is typical to utilize some traditional project management tools, such as MS Project. They were utilized mainly by the project managers, and the developers were not familiar with them. This is mainly due to the space system development process defined in the ECSS standards, which steers the project to be more waterfall development than agile. *Different tools and tools ownership* between the management and developers endangers the customer collaboration, since it steers the communication to be through the project manager. When only the project manager has overall understanding of the project, the interaction between the team members is also endangered.

The number of the space accepted components differs highly from other embedded system areas hindering usage of the latest technologies and components. Also, the standards include quite heavy processes, if the decisions made in earlier milestones are changed. *The difficulty of making changes* challenges the agile value of welcoming changes and may result to not changing the plan to a better one due to the restrictions of the components and process.

Agile software development assumes that the requirements changed and can be even dropped when the deadline arrives. In space system development, the requirements do not change significantly, since they are already agreed in the tender. Instead, the requirements can be discussed with the customer, if they are not seen feasible. It is indeed typical for the space projects to be late—instead of the requirements, it is *the schedule that bends*.

Even though most of the process-related challenges rise from the space system development, the embedded system development also had its effect on some of them. The delivery time of the hardware components plays an important role in the requirement of upfront planning. Due to the slow nature of hardware development, making changes is slower than in software development, but the characteristics of the space system development even adds more challenges in this area.

4.1.3 Case company challenges

The case companies had each their own situations and challenges, which were seen as a motivation for changing the ways of working. In all the case companies, the communication between the team members was based on sitting next to each other and communicating informally. The meetings were focusing more on the schedule and budgets than on the implementation work. There was seen a need for better communication. The information flow in the companies of cases A and B was seen not to be as good as possible, especially with those who worked only for a small portion of their time in the project. In the smallest company, where case C was conducted, the communication challenge was seen to be realizing more in the future, when the company hopefully expands and one person cannot manage all the projects anymore. In agile terms, better team self-organization was hoped for.

4.2 Agile practices as solutions to challenges

Some of the challenges were tied closely to the standards and formal customer interface, which could not be altered in the scope of this project. Instead, the focus was inside the team and how the team could utilize agile practices to

figure out solutions to the challenges. The practices utilized and their connection to the specific challenges are presented in Table 5 and elaborated in this section.

In the core of agile development is the self-organizing team, which organizes its work the best possible way. To support the self-organization, also other agile practices, many adopted from Scrum were taken into use: iterations, planning and reviewing in the beginning and end of the iterations, and frequent status meetings. To help the organization of work, backlogs were utilized, and to support the development of a new working method, retrospectives were arranged in the end of every iteration. A product owner was the main customer contact and also responsible for the backlog.

Iterative development was seen as a way to turn the focus of the work from the individuals to the team. Instead of milestones which last typically several months, the work of the team is inspected in 2- to 4-week iterations giving the team a view of the progress of a product and shorter-term goals. Iterations and their organization give also a possibility to take into account the needs of the project at each point of time: sometimes, the amount of work for the iteration is small and focus can be in another project, or the iteration can be canceled, if it is seen better to continue the development later on.

Planning the work of each iteration as a team gives a view of the interdependencies between the tasks and understanding over each other's work. This may reduce the specialization, at least in the long run, when the expertise is more shared. The required customer documentation can be taken into account in each iteration as work tasks instead of focusing to it in the end of the milestones. The work amount for each developer in each project can be taken into account tackling the challenge of multiple projects. *Reviewing* the product in the end of iterations gives both the customer, if present, and the part-time developers a view of the current status of the product. This also enables team members to share and build knowledge of different areas hopefully reducing the specialization. *Frequent status meetings* focus on knowledge sharing between specialization areas as well as changing the emphasis from the individuals to the team. They offer a place to exchange information and understand the interdependencies between the tasks also in the middle of the iteration.

Backlogs are utilized as a view of the progress of the product supporting the iterations, planning, and review practices. They are targeted for solving the challenge of different tools and their ownership. *Product owner* has a high impact on the formal customer interface, and it is his responsibility to make sure that the backlog is updated according to the customer feedback.

4.3 Experiences of the tailored agile practices

Even though the researchers brought issues into discussion and even suggested solutions, it was up to the teams to decide and define the utilized practices. In all three companies, the backbone of the new process was similar including iterations, planning and review of each iteration, frequent status meetings during iterations, retrospectives for developing further the practices, and backlogs for task allocation.

4.3.1 Practice experiences

Each company and team had their own ways of utilizing the agile practices as was observed by the researchers. These similarities and differences tell how to tailor the practices to the embedded space system development. The end survey, which was conducted after the cases, included a questionnaire about each practice utilized and shared light to the experiences of these practices. In addition to Likert scale questions about usage, potentiality, usefulness, and difference to former practices, the respondents had a possibility to share their thoughts about each practice and how it changed the ways of working.

Iteration. The iteration length was 2 weeks in cases A and B and 3 weeks in case C. Even though the length was discussed in the retrospectives, it was not changed during the pilots.

The initial decision about the iteration length in case A, i.e., 2 weeks, was due to the small project length, and this way, there was time to go through effectively several iterations. The amount of the work to be accomplished and the availability of the resources for the project was considered in the beginning of each iteration. Sometimes the iteration

Table 5 Practices related to the challenges

	Iteration	Planning	Review	Frequent status meetings	Product owner	Backlogs
Specialization of team members		x	x	x		
Formal customer interface			x		x	
Required customer documentation		x				
Multiple simultaneous projects		x				
Emphasis on individual performance	x			x		
Different tools and tool ownership						x

was canceled or prolonged to two iterations length, if the team members were busy with other projects or there was a quieter time in this project. In the end of the pilot of case A, there was a decision to reconsider the length, and for the next project, it was decided to be prolonged even up to 3 months. This was due to impossibility to create concrete targets for the short 2-week iterations.

In case C, the correct length was considered to be 3 to 4 weeks also in the end of the project—similar to what was utilized during the project. This way, the team was able to define the tasks clearly in the beginning of each iteration, even though sometimes the 3 weeks was considered quite a short time, since the project had not proceeded due to the time spent implementing other projects.

According to the end survey, the iterations were seen to create a schedule for the tasks and short common goals making implementation work clearer. This way, the tasks were considered before the implementation. The short iterations and the reoccurring meetings gave rhythm to the work. The iterations helped to prioritize own work and gave understanding also over the work of other developers.

Planning meeting. The basis for the planning meeting was similar in all the three teams. The project manager (acting as a product owner), who was responsible for the project backlog, had already defined the work in higher level, and it was gone through. The team then participated into dividing the work into tasks and giving their opinion not only about the prioritization but also about their current workload. This way, the tasks were selected into each team member and inserted to the iteration backlog.

In case A, the planning meeting consisted of discussion about the work for the next iteration. The product owner picked from the discussion the missing tasks, which were either forgotten to be written or came as new when learning what other team members were about to implement.

In case B, each developer had already before the planning defined tasks, which were considered to be included in the iteration. In the planning, the tasks were gone through and it was decided how much of them could be implemented in the iteration timeframe. A challenge in the planning meetings was that they tended sometimes to include great amount of technical discussion, which was not relevant to everyone and prolonged the meeting.

In case C, the product owner had been already defining the work for the iteration formed tasks from them according to the information he had received from other team members. In planning, the view of the work to be done in the iteration was presented first. Then, every work item and its tasks were discussed through, adding, removing, and changing the original tasks when necessary. The team members gave insight on the scheduling of the tasks during each iteration, enabling taking interdependencies between tasks into account. Also, some new tasks were added to the backlog, which were not supposed to be implemented in this iteration but were utilized as a reminder for the next planning.

According to the end survey, the planning meeting was considered to give a possibility to discuss about the technical solutions. It gave the team a possibility to affect the prioritization of tasks and to understand how the tasks were dependent of each other. Going through the plans together made everyone to think better the tasks to be implemented. The prioritization of the work was defined together, which helped to prioritize own work from the whole product point of view. Also, the amount of work of each developer in each project was taken into account better.

Review meeting. The review meeting consisted basically only going through the tasks and their statuses. Each member of the team had not only the possibility but also the responsibility to tell about the tasks he had finished or had been working on during the iteration.

In case A, especially in the beginning, the content of a task was often too large but was improved during the pilot project. In the review meeting, it was often noted that the definition of done was not clear and the developer thought that the task was not completed even though it was ready from the product owner's point of view. In the iteration, typically only 30–60 % of the tasks were ready in the end of the iteration. This was considered to be due to other higher prioritized projects taking time, the slow nature of the tasks, and the time spent waiting for other tasks to be finished or decisions by the customer.

In case B, in the beginning, each team member took their turn in going through the tasks. Later in the project, the tasks were gone through in work package order, i.e., tasks related to each other were following each other. Also in case B, the amount of completed tasks was about the same level as in case A. Sometimes the low amount of completed tasks was due to other projects suddenly taking more time than anticipated.

In case C, the main target of the iteration was gone through first and then the tasks were gone through in the order they appeared in the backlog. Sometimes when the team members told about their solutions, other team members presented new ideas for the next iteration. The amount of the completed tasks was typically well below 100 %—often other projects had taken more time than anticipated. By the end of the pilot, the review and planning started to unify into one meeting without clear transformation from one part to the other.

According to the end survey, going through the achievements of the short-term goals was more active than before. On the other hand, sometimes the focus was too much in the small tasks and how they were implemented

instead of the overall situation of the product. The visibility of what was planned versus what was actually implemented was seen useful.

Frequent status meetings. The frequent status meetings were quick meetings with the help of the iteration backlog. The status of the project was visualized and the progress of the tasks were gone though, making it possible to continue the technical discussion after the meeting with relevant people. In cases A and B, these meetings were twice a week (once when planning and review were in the same week), and in case B, once a week (except for the week of planning and review).

In case A, some specialists who were not actively participating the project at the time skipped the frequent status meetings. This way, they could diminish the amount of time spent in meetings, when they were implementing several projects simultaneously. The status meetings were also changed from formal meetings to coffee breaks during the pilot project, in order to make them more relaxed.

According to the survey, the frequent status meetings were seen as a good practice to regularly update what had been done and what should be done next. Even though these meetings were seen as a reminder of how the project is proceeding and enablers for the technical discussion, they were seen somewhat useless when everyone had their distinct tasks without interdependencies to each other.

Retrospective and facilitator. The new process was reflected in retrospective meetings after every iteration. In cases A and B, the retrospectives were often going systematically through what was considered good, bad, and delta (i.e., things to be changed) in current practices. Another practice, which was utilized to support the practice evolution, was the role of a facilitator. The facilitator in each team was a member of a team, who was not the lead of the team but was in charge of the working methods similarly to a scrummaster in Scrum.

In case A, even though the challenges were openly discussed and even solutions thought, there seldom was clear improvements to be implemented and followed in the next iteration. Pair and group work was utilized in order to get opinions also from the quieter people. The challenges seemed to be similar during the whole pilot concentrating on how to get a better view of the progress of the product. In case B, the retrospectives were mainly open discussions about the utilized practices and lasted from 20 min to 1 h. In case C, only in the first iteration a clear discussion about the working practices was held. Later on, the retrospective was not implemented.

According to the survey, the retrospectives were seen to aid the process evolution especially in the beginning of the project. On the other hand, when the new process was beginning to stabilize, they were seen to be too often, when utilized in every iteration. According to the end interviews, especially in cases A and B, there was seen a possibility to affect the working methods, but it was not much utilized and the responsibility of the development of the practices was mainly left to the team leaders and the facilitators. In case C, retrospectives were not arranged as such, except in the end of the first iteration. There is intention to organize a project retrospective in the end of the development project.

Backlogs and backlog tools. During the pilot, each company selected their own backlog tools in order to find appropriate tools for the company culture.

In case A, the product backlog was located in Excel and it was derived from the schedule in MS Project. The iteration backlog was a physical whiteboard, where the tasks were printed from Excel. In the beginning, the whiteboard was similar to a Kanban board consisting of to-do, doing, and done, and each project had their own boards. During the pilot project, the board was changed to represent every day of the iteration and the tasks were allocated on a daily basis for each developer, in order to better visualize the amount of work spent in each project during the iteration. When the task was finished, the task was turned upside-down for visualization of the progress.

In case B, the product backlog and iteration backlog were in Redmine [36], which is an issue-tracking tool. The tool was already in use in the project, but now, its usage was systematized and agreed with all the team members. In the beginning, the backlog was formed from what was already planned, but later on, it was based on the work packages. It was also agreed that tasks could be added during iterations, if they were seen critical. This way, all the work would be visible in the backlog.

In case C, the product backlog was a PowerPoint presentation derived from the MS Project, whereas the iteration backlog tasks were held in Excel for each iteration. The product backlog was based on the work packages.

The backlog tools were different in different cases. In general, the utilization of backlog tools was seen useful as a way to visualize and split the work into tasks before the implementation.

4.3.2 Overall benefits and drawbacks

The end survey and interviews focused on the changes in the working methods. The surveys were conducted in cases B and C within the teams, whereas in case A, the opportunity was given to all the employees to answer to the questions. All the respondents in case A were either taking part in the pilot project or in a tendering, which also had started to utilize similar agile practices as the

pilots. Also in the end interviews, different roles were interviewed: project managers, developers, and facilitators. The end interview and survey data are summarized in Table 6.

The answers of the same respondents in case A and case B were compared with Wilcoxon signed-rank test [37]. In case C, there were only three same respondents, which made the statistical comparing impossible, and only visual comparison was made. The smaller number of respondents in case B compared to case A also affected the results: there were more statistically significant changes found in case A than in case B. The found changes with statistical significance ($p < 0.05$) were considered in the interviews whether they were due to the new working methods or something else. Those due to the working methods are presented in Figs. 1, 2, 3, 4, 5, 6, 7, and 8. The interview data also gave deeper understanding over the effects of the new agile practices.

Similar documentation working better. The documentation, which is not seen to be in a focus in agile development, plays a major role in space system development. There were no changes in the actual documentation, apart from the formation of the backlogs. The interviewees also thought that the actual documentation had not changed from the initial situation. Still, in case A, the developers considered that their own work was documented better than before, as can be seen in Fig. 1. According to the interviews, this might be due to the improved communication and understanding over what needs to be documented. In case B, the backlog tool itself was utilized as a temporary space for documentation before fulfilling the official customer required documentation.

Meetings giving rhythm to the development work. The amount of time spent in the meetings was slightly increased in all three companies, and the amount of the meetings was seen to be adequate, as seen in Figs. 2 and 3. According to the interviews, the meetings were also seen worth the time spent. The focus in the meetings was considered to be more on the work at hand, instead of financial and schedule information as had been earlier in the project meetings. This enabled more technical discussion. Especially in the two smaller companies, the teams felt that the regular and systematic meetings enhanced the visibility over the project, which sometimes seemed to proceed quite slowly.

Table 6 End interview and survey data

	Case A	Case B	Case C
End survey response rate	64 % (16/24)	100 % (6/6)	100 % (4/4)
Number of same respondents in initial and end survey	15	6	3
Number of end interviewees	5	2	3

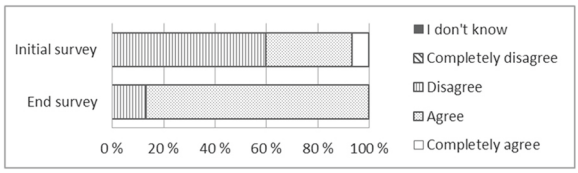

Fig. 1 Case A: I document my own work enough ($p = 0.034$)

Better communication and knowledge sharing. The new practices were seen to enhance the communication and knowledge sharing. Also, those developers, who were working only part-time in the project, were able to participate better in the project. Presenting both the work accomplished and the initiated but incomplete tasks regularly increased the understanding over the interdependencies between the work of each developer. The developers considered it easier to discuss the technical solutions also outside the meetings, when everyone had a general understanding of the work in progress. According to the interviews, the improved communication and knowledge sharing were considered as the major benefits of the utilized agile practices.

Enhanced teamwork. According to the interviews and the survey results, the teamwork had enhanced in all three cases. Besides sharing the results of own work and forming a closer project team, the teams considered to have a possibility to understand and affect the prioritization of work better than before. In case C, all the interviewees agreed that the team targets were now better defined together. In case B, the survey already gave hints over the better allocation of the work between the team members and decreased waiting for other people and their work. According to the interviews, the work amount of other projects was especially taken into account better than before when planning the work together. In case A, understanding of what was currently under work for other team members increased, as presented in Fig. 4.

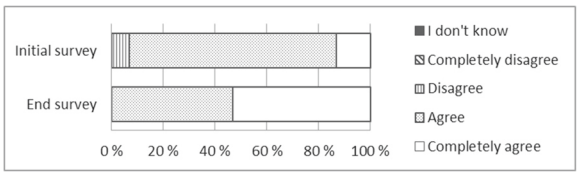

Fig. 2 Case A: There are enough regular project team meetings ($p = 0.008$)

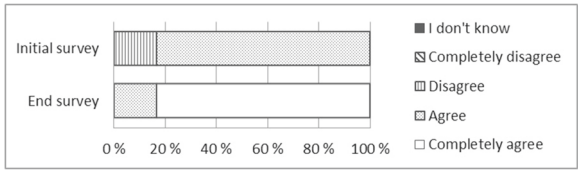

Fig. 3 Case B: There are enough regular project team meetings ($p = 0.034$)

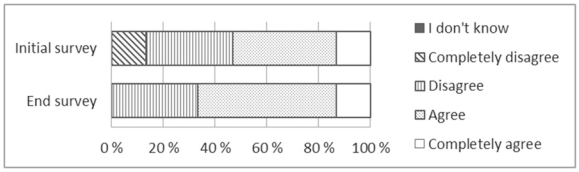

Fig. 5 Case A: The size of the project is easy to understand ($p = 0.046$)

Better view of the project. One of the challenges the teams faced during the project was the view of the whole project. Especially in case A, this was brought up multiple times in the retrospectives: the visualization of the connections and interdependencies between the tasks as well as their connections to the whole project was not considered to be sufficient. Still, there was seen a slight improvement on understanding the size of a project, which can be seen in Fig. 5. Also, the testing could be considered better during the implementation, as presented in Fig. 6. In case B and case C especially, dividing the long project to smaller iterations was seen as a beneficial improvement.

Feedback changes. In case A, the teams experienced an improvement in feedback as presented in Figs. 7 and 8. The new agile practices were seen to be means of finding ways to work more efficiently and going through the feedback of the team. Even though the direct feedback was similar to what it had been, going through the work together was considered as a one way of giving and getting feedback.

Found changes not related to the agile practices. All the changes in the surveys were not related to the changes in the process. In case A, statistically significant changes of this kind could be found. Even though the documentation and specifications had improved from the re-use point of view, it was also seen to be due to the project situation—one of the pilot projects was focused on the re-use of the design. Also, there was seen a better attempt to cycle the tasks between the team members, but this was partly due to other company improvements: independent of this project, the company was focusing on having multiple persons being able to implement any development task. The requirements were not seen as strict as before, which was more related to the projects at hand than in the working methods.

In other two companies, some small, but not statistically significant differences could be found from the survey. In case B, the documentation required by the customer was not anymore enough for the personal need, but the reason was seen to be that the current customer did not have as heavy or thorough documentation requirements as the customers during the initial survey. In case C, the team experienced more waiting for others and also other projects disturbing the implementation of the pilot projects, but these two changes were seen to be related to the project situation, where the delays of the project were customer originated.

5 Discussion

In the research, several challenges for utilization of agile methods in embedded space system development were recognized. On the other hand, agile practices were also able to help to conquer some of these challenges. The challenges found are such that most of them can be found in other settings as well, but according to the case studies, they were apparent in all of these three cases and can thus be argued to be embedded space system development challenges in SMEs.

The specialization was seen as one of the challenges, and the main reason was the special environment, where the space systems must function. In software or other embedded system development, the amount of different specialist is usually smaller, but there may exist environments requiring similarly multiple specialization areas. In these kind of environments, the similar aspects of agile practices can be also fruitful: the utilization of planning,

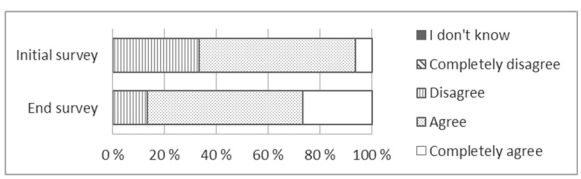

Fig. 4 Case A: I know all the time what other developers in the same team are working on ($p = 0.034$)

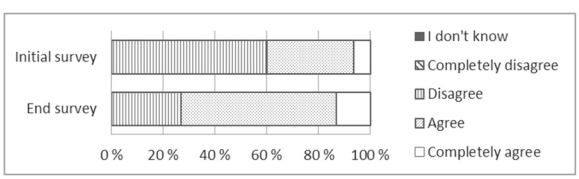

Fig. 6 Case A: Testing and verification are integral parts of the product development process ($p = 0.014$)

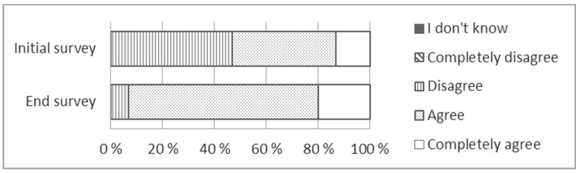

Fig. 7 Case A: We are continuously looking for ways to work more productively and effectively ($p = 0.035$)

review, and frequent status meetings as a way to share the knowledge between the team members and enablers to take into account the interdependencies between the tasks. The benefits from the practices were evident: even though everyone still had their special areas in the projects, the specialization was not any more a relevant issue from the product progress point of view. The documentation remained the same, but the communication was enhanced.

The emphasis of individual performance was due to the specialization, and also, the long milestones were seen as the reason for working alone for a long time without frequent feedback of the work. Iterations were seen helpful as giving shorter goals and pace to the development work and made the implementation work clearer. In other words, the emphasis was changed from the individuals to the teams, when the whole team understood what was going on in the project at any given time.

Multiple simultaneous projects were considered especially in the planning, when taking care of the workload of each developer and iterations could be even canceled. In case A, the multiple projects were even visualized with the physical board. This gave a better view of what had been achieved in this project and how other projects influence the progress of this project. Still, developing multiple projects at the same time was seen as a burden. While agile methods assume that the team can focus on one project at the time, similar situations can be faced due to various reasons also in software and embedded system development and also similar solutions can be utilized.

Clearly, the challenge of different *tools and tools ownership* is something that is present everywhere, where the development is more waterfall-based. In space system development, it was due to the standards that steered the development towards the more waterfall model. Thus, it is quite straightforward also in other environments to utilize backlogs in some form in order to get the team collaborate and interact better inside the team and with the customer.

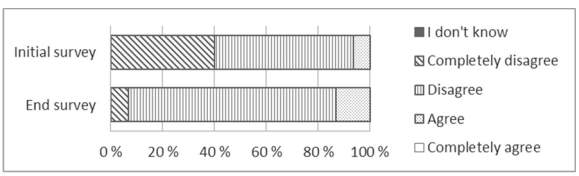

Fig. 8 Case A: Team feedback is reviewed systematically ($p = 0.034$)

Formal customer interface was tackled with a product owner. Even though the project manager, who mostly acted as a product owner, shared the responsibility of the project better with the team, he was still acting similarly as the main customer contact point. *The required customer documentation* was taken into account in planning meeting, where documentation tasks were created to ensure that the documentation was produced simultaneously with the implementation. In areas of safety-critical environments, formal customer interface and the amount of required documentation are usually based on the standards and their interpretations. Thus, when utilizing agile methods in standardized environments, the backlogs and the product owner can be seen as a key to solve these challenges. To tackle better the challenges of multiple simultaneous projects and formal customer interface, the customer should be more involved with the project according to agile values. Then, there might be a possibility to focus for a shorter time on one project and get quicker customer feedback to direct the product development to the correct direction.

Enhancing the customer collaboration could also help the two other challenges identified: *difficulty of making changes and schedule bending instead of requirements*. With better collaboration, making changes could be easier both ways—to the requirements and to the implementation, even though the formal documentation would be updated a little later. Since detailed upfront planning is required by the standards, the standards would need alteration. Still, in the end, the space reliability requirements will affect component selection and general nature of combined hardware and software development will require some amount of *upfront planning*.

Even though agile methods could not be utilized literally, it was seen that in the team level, the benefits were undeniable. The drawbacks could not be seen, but the follow-up time was still quite short and none of the projects were in the critical path for the customer. This allowed spending more time in experimenting the new ways of working. Already in the few months follow-up time, the new practices were seen to have notable benefits, and all the teams were motivated to utilize and further develop their new agile working method.

5.1 Validity and reliability

In order to improve validity and reliability, three case studies in three different companies were utilized. This improves the relevance of the findings regarding the whole embedded space system development. The three companies represented quite well the space sector in small

European countries: they were small companies, of which only one was focusing solely in space sector.

The researchers took an effort to participate into the process-defined meetings at least once in every iteration for each company in order to gain understanding over the work and how the practices influenced it. This way, also the understanding between the company representatives and researchers improved diminishing the possibility of misunderstandings.

From the reliability point of view, multiple sources of data were utilized, and the analysis was considered always by several researchers to avoid researcher bias. In addition, the results were openly discussed with team members, in order to gain confidence on the interpretations by the researchers.

6 Conclusions

Agile methods are seen to improve efficiency and productivity without endangering well-being at work in software development, where they originate from. These similar benefits are desired in any industry, and the challenges, the ways to tailor the practices, and the benefits in utilization of the practices in embedded space system development were researched through a case study in three different companies.

In the first phase, the challenges the embedded space system development poses to agile methods were researched through a survey about the working methods for all the case company employees and interviews for different roles in the case companies. In the second phase of the research, the utilization of the agile practices was followed in the pilots of the three companies. In the last phase, the survey was conducted again with additional questions about the utilized practices. Also, a set of interviews was conducted in order to understand whether the changes were due to the new agile practices or other changes and in order to give better understanding over the experiences.

The challenges found were factored to the agile values which they contradicted. One of the processrelated challenges was the formal customer interface with required extensive customer documentation. This challenge endangers *customer collaboration* and focusing on *the working product*, which are two of the agile values. Detailed upfront planning contradicts with the nature of *iterative and incremental development* as well as transfers the focus from working product to the plans and makes changes difficult. *Making changes* even late in the development is endangered also by the slow and heavy processes for making changes, and thus, instead of the requirements, it is often the schedule that changes. When utilizing traditional project management tools, the tool ownership is usually in the management side and endangers the work as the a self-organizing team and hinders customer collaboration. The teamrelated challenges were the high specialization of the team members, development of multiple projects simultaneously, and emphasis on individual performance, which all highlight individuals but diminish *the interactions* between team members.

Agile practices were seen as solutions to the challenges found, especially to those related to the teams. The teamwork was enhanced in planning and reviewing the work together in short iterations transferring emphasis from the individuals to the team. Multiple projects and the slow nature of projects were taken into account, e.g., in defining the amount of frequent status meetings only once or twice a week and considering the current amount of work for the project in the planning meetings; sometimes the iteration could be even canceled. All the meetings were seen as enablers for the enhanced technical discussion, which steered the product into the correct direction. The backlogs enhanced the visualization of the progress of the product.

Most of the challenges found are not unique only in the embedded space system development but can be present in other environments as well: especially the challenge of *tool ownership*, which comes from the waterfall-like development, and also the challenges related to standards and safety-criticality such as detailed upfront planning, difficulty of making changes, and required customer documentation. In these kind of environments, agile practices may help the development work similarly to the embedded space system domain.

This case study showed that agile practices are useful inside a subcontractor project team in embedded space system development, even though the rest of the development is utilizing more traditional methods. The most focal benefits experienced by the teams were (1) the improved communication through more efficient meeting routines, (2) the improved teamwork through planning the work together as a team instead of defined by the project manager, and (3) the transparent rhythm to the sometimes slowly proceeding embedded space system development work through utilization of iterations.

The case study also showed that several challenges were related to the customer interface, which was not changed in the scope of this research. Thus, it would be beneficial in future research to alter the working methods of a customer-subcontractor chain to see whether more benefits could be received, such as better productivity and keeping the original schedule.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The research reported in this article has been conducted as a part of the AgiSpacES (Agile Development Methods for Embedded Systems in Space

Industry) project. The project is carried out in collaboration with industry partners RUAG Space Finland, AL Safety Design, Aboa Space Research, Harp Technologies and Kovilta. The project is mainly funded by Tekes – the Finnish Funding Agency for Technology and Innovation.

Received: 23 December 2015 Accepted: 23 June 2016
Published online: 07 July 2016

References

1. T Dybå, T Dingsøyr, Empirical studies of agile software development: a systematic review. Inf Softw Technol. **50**(9-10), 833–859 (2008). doi:10.1016/j.infsof.2008.01.006
2. P Rodríguez, J Markkula, M Oivo, K Turula, in *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM '12*. Survey on agile and lean usage in Finnish software industry (ACM, New York, NY, USA, 2012), pp. 139–148. doi:10.1145/2372251.2372275. http://doi.acm.org/10.1145/2372251.2372275
3. VersionOne, The 8th Annual State of Agile Development Survey (2013). http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf. Accessed 20 June 2016
4. 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, RC Martin, S Mellor, K Schwaber, J Sutherland, D Thomas, Manifesto for Agile Software Development. http://www.agilemanifesto.org/. Accessed: August 18, 2015
5. K Schwaber, M Beedle, *Agile Software Development with Scrum*, 1st edn. (Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001)
6. K Beck, C Andres, *Extreme Programming Explained: Embrace Change*, 2Nd Edition. (Addison-Wesley Professional, Boston, MA, USA, 2004)
7. ESA, The European Space Agency. http://www.esa.int. Accessed 20 Dec 2015
8. ECSS, Collaboration Website of the European Cooperation for Space Standardization. http://www.ecss.nl/. Accessed: September 22, 2015
9. ECSS, Space project management: project planning and implementation. http://www.ecss.nl/. Accessed: September 22, 2015
10. C Larman, *Agile and Iterative Development: A Manager's Guide. Agile software development series*. (Addison-Wesley, Boston, MA, USA, 2004). https://books.google.fi/books?id=e4FrAFn0ytIC
11. C Larman, VR Basili, Iterative and Incremental Development: a Brief History. Computer. **36**(6), 47–56 (2003). doi:10.1109/MC.2003.1204375
12. O Salo, P Abrahamsson, Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. Softw IET. **2**(1), 58–64 (2008). doi:10.1049/iet-sen:20070038
13. M Laanti, in *System Sciences (HICSS), 2013 46th Hawaii International Conference On*. Agile and wellbeing—stress, empowerment, and performance in scrum and kanban teams (IEEE Computer Society, Washington, DC, USA, 2013), pp. 4761–4770. doi:10.1109/HICSS.2013.74
14. ESA, The ESA Concurrent Design Facility—Concurrent Engineering Applied to Space Mission Assessments (2015). http://esamultimedia.esa.int/docs/cdf/CDF_infopack_2015.pdf. Accessed: September 22, 2015
15. SE Carpenter, A Dagnino, in *Space Mission Challenges for Information Technology (SMC-IT), 2014 IEEE International Conference On*. Is agile too fragile for space-based systems engineering? (IEEE Computer Society, Washington, DC, USA, 2014), pp. 38–45. doi:10.1109/SMC-IT.2014.13
16. R Santos, F Flentge, M-E Begin, V Navarro, in *Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing*, ed. by A Sillitti, O Hazzan, E Bache, and X Albaladejo. Agile technical management of industrial contracts: scrum development of ground segment software at the european space agency, vol. 77 (Springer, Berlin Heidelberg, 2011), pp. 290–305. doi:10.1007/978-3-642-20677-1_21. http://dx.doi.org/10.1007/978-3-642-20677-1_21
17. E Ahmad, B Raza, R Feldt, T Nordebäck, in *Proceedings of the 2010 National Software Engineering Conference. NSEC '10*. ECSS standard compliant agile software development: an industrial case study (ACM, New York, NY, USA, 2010), pp. 6–166. doi:10.1145/1890810.1890816. http://doi.acm.org/10.1145/1890810.1890816
18. B Fitzgerald, K-J Stol, R O'Sullivan, D O'Brien, in *Proceedings of the 2013 International Conference on Software Engineering. ICSE '13*. Scaling agile methods to regulated environments: an industry case study (IEEE Press, Piscataway, NJ, USA, 2013), pp. 863–872. http://dl.acm.org/citation.cfm?id=2486788.2486906
19. A Sidky, J Arthur, in *Proceedings of the 31st IEEE Software Engineering Workshop. SEW '07*. Determining the applicability of agile practices to mission and life-critical systems (IEEE Computer Society, Washington, DC, USA, 2007), pp. 3–12. doi:10.1109/SEW.2007.99. http://dx.doi.org/10.1109/SEW.2007.99
20. R Hoda, P Kruchten, J Noble, S Marshall, in *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications. OOPSLA '10*. Agility in context (ACM, New York, NY, USA, 2010), pp. 74–88. doi:10.1145/1869459.1869467. http://doi.acm.org/10.1145/1869459.1869467
21. 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 J. Embedded Syst. **2013**(1) (2013). doi:10.1186/1687-3963-2013-15
22. PM Huang, AG Darrin, AA Knuth, in *Aerospace Conference, 2012 IEEE*. Agile hardware and software system engineering for innovation (IEEE Computer Society, Washington, DC, USA, 2012), pp. 1–10. doi:10.1109/AERO.2012.6187425
23. D Morgan, Covert agile: development at the speed of. AGILE Conf. **0**, 79–83 (2009). doi:10.1109/AGILE.2009.48
24. A Shatil, O Hazzan, Y Dubinsky, in *Software Science, Technology and Engineering (SWSTE), 2010 IEEE International Conference On*. Agility in a large-scale system engineering project: a case-study of an advanced communication system project (IEEE Computer Society, Washington, DC, USA, 2010), pp. 47–54. doi:10.1109/SwSTE.2010.18
25. E Gul, T Sekerci, AC Yücetürk, N Yildirim, in *ICSEA*. Using XP in telecommunication software development (IEEE Computer Society, Washington, DC, USA, 2008), pp. 258–263
26. J Drobka, D Noftz, R Raghu, Piloting XP on four mission-critical projects. IEEE Softw. **21**(6), 70–75 (2004). doi:10.1109/MS.2004.47
27. NV Schooenderwoert, in *AGILE 2006 (AGILE'06)*. Embedded agile project by the numbers with newbies (IEEE Computer Society, Washington, DC, USA, 2006), pp. 13–366. doi:10.1109/AGILE.2006.24
28. DD Santos, NID Silva, R Modugno, H Pazelli, A Castellar, in *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, ed. by K Elleithy. Software development using an agile approach for satellite camera ground support equipment (Springer, Dordrecht, 2007), pp. 71–76
29. B Greene, in *Proceedings of the Agile Development Conference. ADC '04*. Agile methods applied to embedded firmware development (IEEE Computer Society, Washington, DC, USA, 2004), pp. 71–77. http://dl.acm.org/citation.cfm?id=1025113.1025144
30. P Manhart, K Schneider, in *Proceedings of the 26th International Conference on Software Engineering. ICSE '04*. Breaking the ice for agile development of embedded software: An industry experience report (IEEE Computer Society, Washington, DC, USA, 2004), pp. 378–386. http://dl.acm.org/citation.cfm?id=998675.999442
31. CE Matthews, in *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11. SPLASH '11 Workshops*. Agile practices in embedded systems (ACM, New York, NY, USA, 2011), pp. 249–250. doi:10.1145/2095050.2095091. http://doi.acm.org/10.1145/2095050.2095091
32. J Ronkainen, P Abrahamsson, in *Extreme Programming and Agile Processes in Software Engineering: 4th International Conference, XP 2003 Genova, Italy, May 25–29, 2003 Proceedings*, ed. by M Marchesi, G Succi. Software development under stringent hardware constraints: do agile methods have a chance? (Springer, Berlin, Heidelberg, 2003), pp. 73–79
33. M Kaisti, T Mujunen, T Mäkilä, V Rantala, T Lehtonen, in *15th International Conference on Agile Software Development XP2014*. Agile principles in the embedded system development (Springer, Cham, 2014)
34. RK Yin, *Case Study Research: Design and Methods*. (SAGE Publications, Thousand Oaks, California, USA, 2009)
35. P Runeson, M Höst, A Rainer, B Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. (Wiley Blackwell, Hoboken, USA, 2012)
36. Redmine - Project management web application. http://www.redmine.org/. Accessed 20 Dec 2015

37. NJ Salkind, K Rasmussen, (eds.), *Encyclopedia of Measurement and Statistics* (Sage Publications, Inc, Thousand Oaks, CA, USA, 2007). doi:10.4135/9781412952644. http://dx.doi.org/10.4135/9781412952644
38. K Könnölä, S Suomi, T Mäkilä, T Jokela, V Rantala, T Lehtonen, Agile methods in embedded system development: Multiple-case study of three industrial cases. Journal of Systems and Software. **118**, 134–150 (2016). doi:10.1016/j.jss.2016.05.001