Challenges in Agile Security Engineering: A Case Study[1]

Kalle Rindell[2], University of Turku
kakrind@utu.fi
Department of Future Technologies
20014 University of Turku
Finland

Sami Hyrynsalmi, Tampere University of Technology
sami.hyrynsalmi@tut.fi
Pervasive Computing
PL 300, 28101 Pori
Finland

Ville Leppänen
ville.leppanen@utu.fi
Department of Future Technologies
20014 University of Turku
Finland

Abstract

This chapter describes a case of a large ICT service provider building a secure identity management system for a government customer. Security concerns are a guiding factor in the design of software-intensive products and services. They also affect the processes of their development. In regulated environments, development of products requires special security for the development processes, product release, maintenance and hosting, and also require security-oriented management and governance. Integrating the security engineering processes into agile development model is argued to have the effect of mitigating the agile methods' intended benefits. The project case was an effort of multi-team, multi-site, security engineering and development work, executed using the Scrum framework and regulated by governmental security standards and guidelines. In this case research, the experiences in combining security engineering with agile development are reported, challenges discussed and certain security enhancements to Scrum are proposed.

Keywords: Security, Standard, VAHTI, Infrastructure, Case Study, Scrum, Agile

[1] This paper extends article "Case Study of Security Development in an Agile Environment: Building Identity Management for a Government Agency" by Rindell, Hyrynsalmi & Leppänen (2016).

[2] Corresponding author

1. Introduction

Security regulations are an important driver in various aspects of software development and information systems and services. Even in the cases when formal security standards or guidelines are not strictly required the drive for security still guides the selection of design patterns and technological components, as well as the design and development work. Increasing diversity in development methods, technology, and the environments where the systems are used, have prompted organizations to follow various security standards, as well as created the need to establish new ones to guarantee adequate security assurance. In 2001, the government of Finland begun to issue a set of security regulations, called VAHTI instructions[3]. Compliance with the instructions is now mandatory for all government agencies and the regulation is also applied to any information system and data connected to a VAHTI-classified system.

While the importance and use of security regulations has increased, the use of lightweight software development processes and methods, i.e., *agile development*, has become the *de facto* standard in the industry (VersionOne, 2016). While there exists a series of suggested methods how to conduct security engineering activities in an agile project (see e.g. Baca & Carlsson, 2011; Beznosov & Kruchten, 2004; Fitzgerald, Stol & Sullivan, 2013; Ge, Paige, Polack & Brooke, 2007; Pietikäinen & Röning, 2014; Rindell, Hyrynsalmi & Leppänen, 2015:1), the empiric evidence is still largely anecdotal and the cases reported specific to an industry or a single company. The study reported in this paper is exploratory, and thus the research, by its nature, explorative. This study reports the experiences in agile development in a security-regulated environment. The research objective (RO) is:

> **RO:** Identify advantages, best practices and the possible disadvantages of using agile software development methodologies in security engineering.

The results contribute to the on-going discussion by being a result of a deep analysis of combining security engineering with an agile method in an industry setting. Furthermore, the result of this study pave the way for further work deepening our understanding on the benefits and drawbacks of using agile software development methodologies in security sensitive development work.

In the case examined, a Scrum project was conducted with the objective of building an IDM system for information systems compliant with the security regulations. This included building a secure compliant server platform to host the IDM; the same platform would be utilized also to host the client's other information systems. Also software development projects' infrastructure would be hosted on the same platform, although with certain (unrelated) dispensations from the client's security regulations.

The project was executed from 2014 to 2015, spanning over 12 months. Depending on the tasks in each sprint, the team was split into two to three

[3] https://www.vahtiohje.fi/web/guest/home

geographically dispersed groups. The client, a government agency that initiated the building of the platform, mandated the project to be managed by unmodified "textbook version" of Scrum. This called for strict adherence to fixed-length sprints, well-communicated product and sprint backlogs and daily progress monitoring by the Product Owner, and the steering group watching over all the ongoing projects. The project was under strict control of the Project Management Office, and schedules of related infrastructure and software development projects were depending on the results of this project. Compliance with the government regulation, VAHTI, was a central objective of the project. In addition to VAHTI, the client agency had also their own additional security demands, as well as recommendations from other government agencies, most importantly the National Cyber Security Centre's (NCSA-FI)[4] security instructions. The server platform to be built was to be acceptable for use for all government agencies, as well as private companies or organizations requiring similar level of VAHTI compliance.

This paper presents how Scrum was applied for the security-related work required in the project, the challenges that were met, and how the project was overall conducted. As observed, not all the objectives of using "pure" Scrum were not met; therefore, suggestions are made to improve the efficiency of the development work by e.g. introducing rudimentary security engineering extensions to the Scrum framework. These extensions include a new role for a security developer. In addition, use of specific security sprints and other security-oriented additions are suggested. We also discuss how the introduction of the security engineering activities into the project affect cost, efficiency and the conduct of the project.

Chapter 2 will present the related work and standards, as well as clarify the motivation for the study. The Scrum process and some useful security augmentations to it are presented in Chapter 3. This is followed by an explanation of the research process, and the description of the case. In Chapter 6, an analysis is presented, and the last two chapters conclude the study with discussion and proposals for future work.

2. Background and motivation

A software development process model depicts how development work is divided into smaller parts and how those are managed. In addition, the method may define different artefacts, tools, ceremonies and practices that should be used. Currently, agile software development methods form the current state of the art in software development projects, whereas the security standards regulating software development processes, such as ISO/IEC 21817 (2008) and ISO/IEC 27002 (2013) originate in the time preceding the agile methods.

---

4 https://www.viestintavirasto.fi/en/cybersecurity/ficorasinformationsecurityservices/ncsa-fi.html

Based on the literature, the typical approach to agile security engineering is to simply start using the methodology at hand without formal adjustments – the observations made in this particular case follow this same pattern. In the literature, notable exceptions of thorough and formal approach to security engineering are described e.g. by Baca & Carlsson (2011) and Fitzgerald & al. (2013). There are even well documented cases of attempts to achieve formal ISO/IEC capability maturity level incorporating agile methods, such as Diaz, Garbajosa & Calvo-Manzano (2009). Unfortunately, the findings and suggestions made in these studies were not directly applicable in a project that was not strictly restricted to software development.

Instead, a more *ad hoc* approach was used by the project team of this case study. In this approach, the security-related tasks are treated simply as any other items in the backlog: the security requirements are converted to tasks, given story points, and completed among the other items as best seen fit. In cases when security items could not reasonably be time-boxed, because of the inherent uncertainties in the work estimate – or the inexperience of the team – they were separated from the Scrum sprint cycle and completed in non-time-boxed spikes. This was a common pattern throughout the project, and a notable motivation to examine if the project could have benefited from the ´divide and conquer' approach enforced by restricting the tasks to the length of a sprint. Although technically spikes conform to the Scrum methodology, this can also be seen as a partial abandonment of agile methods when performing the crucial security tasks of the project.

While the ad hoc method may succeed in achieving "minimum viable security" by complying with the formal requirements, it is hardly the most effective way to achieve the goals, nor does it necessarily provide the best security assurance for the product. Producing proper security assurance is possible with careful planning, hindered by a lack of proper security requirement management and security task pre-planning. Absence of these elements in the project management methodology tend to lead to inefficiencies and consequently delays, as well as increased development costs. Lack of proper security assurance may also increase the amount and severity of the residual security risk during the software system's life span.

Our argument is that by adjusting the Scrum methodology to better align with the goals of security engineering, the security cost overhead can be reduced while the security of the end product is enhanced, compared to traditional sequential security engineering practices. This is achieved by incorporating the security processes into Scrum activities, as opposed to treating them merely as items in the backlog, by introducing new security-oriented roles into the development team. By incorporating the security engineering activities into the development method, the full benefit of incremental agile methods can be utilized to achieve better efficiency ratio and, arguably, better software products.

The next subsections provide more information about the used governmental security standard, VAHTI, and the use of Scrum methodology in development

projects requiring security standards compliance as well as the related work. Notable similarities in the software security and safety regulations, and the ways they are enforced, prompt a suggestion that similar methodologies are applicable to both fields of requirement.

2.1. Related work

To the best of the authors' knowledge, this work is among the first empirical explorations on an industrial setting where security engineering project has been handled with an agile method. While, e.g., Boström et al. (2006) studied empirically on the applicability of secure enhanced XP method, they used students in their controlled experiments. However, due to the simulative nature of the student cases as well as limitations and imbalance of students' skills, the applicability and generalizability of results are limited, at the best.

Despite the lack of empirical evidence, a series of different methods and adaptions to the existing processes have been presented. In 2004, only a few years after the agile manifesto, Wäyrynen et al. (2004) discussed on the applicability of and needed improvements to XP development method to a security engineering project. Kongsli (2011) presented an approach to integrate security mechanisms into the agile methods. Chivers, Paige and Ge (2005) propose the use of iterative security architecture for achieving agile security. They continued the work in Ge et al (2007) by proposing security practices for XP.

While this research concentrates more on the overview of the issues faced in a project including a considerable amount of security work, also more detailed descriptions of security work have been created. Villamizar et al. (2018) have conducted a systematic mapping of security requirements engineering in agile development. They identify several approaches to meet these security requirements. Among them are process adjustments, introducing new security-related artefacts and guidelines to handle security issues – all of which were also observed in the project reported in this article. Cruzes et al. (2017) provide detailed empiric reports on security testing in agile development, suggesting further awareness on security issues to improve the effectiveness and coverage of security testing. Morrison et al. (2017) have studied how well development teams adhere to security procedures, also measuring the security engineering based on the time spent on security during development. Their findings are partially consistent with the observations in this study, particularly on the strong negative experienced effect of security tooling and security reviews on the work schedule. Contrary to this, security documentation of the technical stack was experienced positively in the study of Morrison et al., whereas our experience reports indicated a clearly negative impact of the extensive documentation requirements.

Achieving software security is a goal separate from software security, although it is achieved by very similar means. Kasauli et al. (2018) have conducted a mapping study of agile development of safety-critical systems, listing also the

challenges found in industry projects. Their findings have a strong resemblance to our observations: waterfall mindset, strong focus on documentation and reported *lack of trust in agile methods* were all evident in our project as well; in practice, the waterfall mindset resulted in abandonment of iterative workflow, a key component of agile methods, in crucial project tasks. Heeager et al. (2018) have completed a systematic literature review of agile development in safety-critical context. They identify four "problem areas" in the research field: documentation, requirements, lifecycle and testing. Although their focus is more on the quality assurance issues, the literature analysis suggests that these areas are interdependent. In security, similarly to safety, the security requirements do not change; however, if agile development models are abandoned because of this, the quality improvement and other benefits of agile methods are lost altogether, adversely affecting the implementation of requirements that *do* change.

Recent works by Othmane et al. (2014), Sonia, Singhal and Banati (2014) as well as by Rindell, Hyrynsalmi, and Leppänen (2015:2) have aimed to more complete approach. For example, Rindell et al. (2015:2) presented an applied version of Scrum method that fulfills the national security development requirements. Yet, their study is also lacking an empirical evidence on the applicability of the presented method. Thus, this study aims to explore a real-world industrial case in order to provide first evidence on the actual applicability of agile methods in security engineering. In the following, we will present Scrum and some of its security enhancements as it was used by our case study team.

3. Security-augmented Scrum

Scrum is a generic agile framework, originally intended to manage software development projects with small co-located teams. Scrum suggests that the product to be completed is divided into smaller components, or features, based on the customer requirements. These requirements are represented by user stories, which are then translated into product features by the development team. Features are then further divided into work packages or items, which are compiled into a product backlog. Items in the product backlog are completed in an incremental and iterative manner during short-term development sprints. The team, consisting of the Scrum Master, the Developers, and the Product Owner as customer's representative, determines the items to be completed during the next 2-4 weeks sprint, consisting of daily scrums. After the sprint, the work is demonstrated, and optionally the team performs self-assessment of the past sprint in a retrospect event.

In this representation, the Scrum process is augmented by three major extensions, presented in Figure 1.

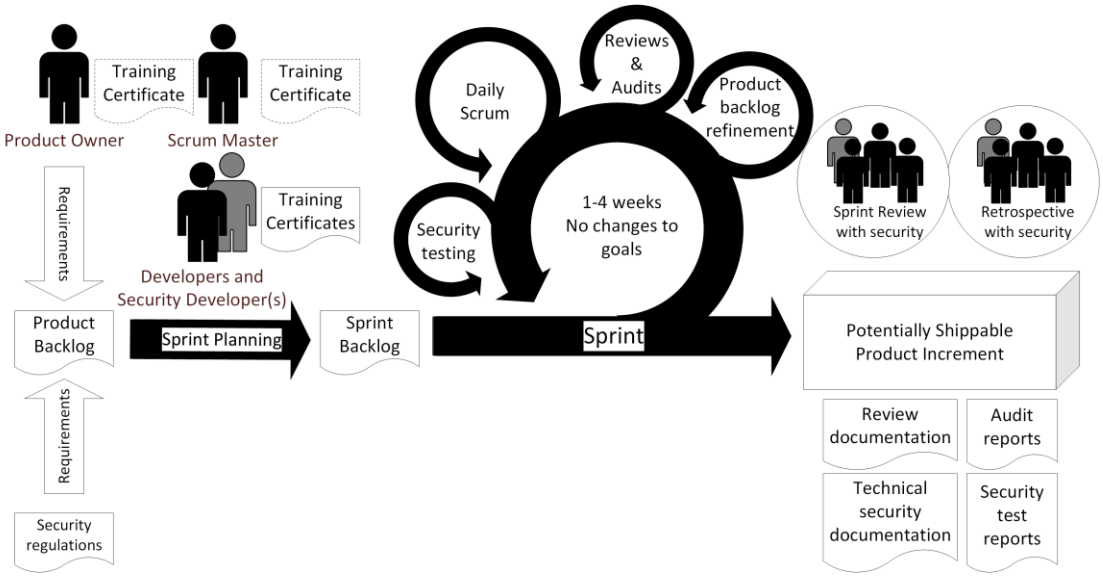

Figure 1. Security-oriented Scrum process and roles (adapted from Rindell, Hyrynsalmi & Leppänen (2015:2).

1. The role of a *security developer*. The security developer, or developers, focus on the security of the product, and typically create or review the documentation required to pass the security audits.
2. Security assurance provided by creating *security artefacts*, mostly security-related documentation. They consist of security training certificates required from the project team, but most importantly the architecture documentation, risk management plans, test plans, test reports, system's log files and other evidence required by the security auditor. The audits also produce reports, which are part of the security assurance provided for the customer.
3. Anticipation of and planning for *security-related tasks*. To better illustrate this aspect of security work, security engineering activities are presented as iterative tasks in the sprint cycle in addition to the daily scrum. It should be noted that not all sprints may have all the security tasks, and if the organization decides to perform security-oriented security sprints, the daily scrum may entirely consist of security activities.

In a project using unmodified Scrum, such as the one used in this case, the security testing, reviews and audits are viewed as normal stories in the sprint backlog and executed as part of the daily scrum. In this view, the security tests and audits are part of the product, as compliance with security standards and regulations is mandatory during development time. The main shortcoming is the difficulty or outright inability to estimate the amount of work involved in the security activities, which merits for giving them special treatment. By emphasizing the importance and special role of the security stories, compared to treating them as overhead and extra burden, is prospected to produce better results with higher efficiency. In effect, this will reduce the cost of the development work.

VAHTI is an open and free collection of the Finnish government's security guidelines, published on the Internet since 2001. The aim of this regulatory

framework is to promote and enforce organizational information security, risk management and overall security competence of various government agencies, and harmonize security practices throughout the organizations. As of spring 2016, the collection comprises of 52 documents. The following VAHTI instructions were found to be relevant for this project:

- VAHTI 2/2009 "Provisions for ICT service interruptions and emergencies", VAHTI (2009:2)
- VAHTI 2b/2012 "Requirements for ICT Contingency Planning" , VAHTI (2012:2b)
- VAHTI 3/2012 "Instructions for Technical Environment Security", VAHTI (2012:3)

Of these, only the document 2b/2012 is available in English. The other relevant documents are made available in Finnish, and their English titles translated for the purpose of this chapter. This also applies to much of the VAHTI terminology: official English translations may not exist, may be inconsistent between documents or may change over time. Even the name of VAHTI board itself has changed at least twice after 2015, although the acronym remains unaltered.

In addition to the VAHTI requirements, the company responsible for building the platform is audited for compliance with ISO/IEC standards 9001, 27001, 27002, and 21817, as well as its own extensive management framework, which it makes available for its clients for review. The company has functions in the United States, so also Sarbanes-Oxley (SOX) act applied. SOX is mostly concerned with the financial elements of the project, but still affected the work load of the Scrum Master by adding certain reporting responsibilities.

VAHTI classifies the information systems into three security levels: basic, increased and high. The server platform, where the IDM system was installed, was built for the increased security level. Information contained in the systems is classified into levels from IV to I, level IV being the lowest. Information contained in a system audited for increased security level may contain clear-text information up to level III. In this case, however, all data stored on the platform is always encrypted despite the official classification level.

According to Hope, McGraw & Anton (2004), software security is an emergent property, not a feature or set of features of the software. The term `security engineering' used in this chapter comprises all security-related *software* engineering tasks within a software-intensive product's life cycle. In current standardization, these activities are categorized into three main process areas: risk, engineering and assurance processes, as presented in ISO/IEC 21817:

S*ecurity risk process* assesses the risk and aims in minimizing it by assessing threats and vulnerabilities, and the impact they have, producing risk information.

Security engineering process uses this information with other security-related input to define security needs and provides solutions to fill them.

Security assurance process collects and produces evidence of security's existence, and aims in its verification and validation.

The ultimate goal of these processes is to identify and mitigate security threats, and define the impact and actions to be taken when the residual or unrecognized risk is realized: what will happen if, and when, the security becomes compromised.

In the Scrum development process both functional (business) requirements and the non-functional (quality, architecture, environment) requirements are transformed into a working software product. Security is typically classified as a non-functional requirement: these are noted to receive lessened attention in the agile methods (see e.g. Ramesh, Cao & Baskerville, 2010). However, in the case of *regulated security*, the distinction between functional and non-functional requirements becomes irrelevant, as incorporation of security requirements is an absolute acceptance criterion for the product. The security requirements were given as strict list of processes to put in place, functionality to implement and artefacts (security assurance) to produce. These requirements were all evaluated, and assigned story points in a iteration planning event; however, as shown in the case description (Chapter 5), this process failed with certain types of security tasks, as they could not be fitted into the rigidly time-boxed iterations. This happened at least partially because the security experts were used to work in a more traditional way, and where the work units (tasks) are considerably larger and, in this case, apparently indivisible. Despite not being part of sprints, the performers of security-related items were still adhering to the Scrum practices, and their work was monitored in daily meetings. In the team's experience, Scrum provided clear and consistent improvement to the security engineering work also in this sense.

4. Research process

This study follows a case study design method by Yin (2003), and a qualitative research approach by Cresswell (2003). The research approach is exploratory as there are only little empirical evidence prior this study, and acts as a research effort towards revealing some of the mechanisms for future inquiries. For the study, a development project utilizing agile methods in compliance with security regulations or standards was sought out. VAHTI regulations, as a national security standard in Finland, provided this context readily in Finland. In addition, a project either already finished or near its ending was favored in order to provide quick access to the challenges and solutions, and to a chance to evaluate the success of the model used in the project. Finally, the selected case should be a representative candidate as well as be able to produce rich information about the phenomenon under study.

In the project case an identity management and verification service was ordered by a governmental client, who also required the use of VAHTI security instructions; also, their acquisition guidelines mandated use of Scrum, providing

an ideal target for research. The development work in the project was done by following a modified version of Scrum software development method. As Scrum is currently one of the most used development methods, the findings from this case study can be held representative of industry practices.

The project was executed by a well-known software product development and consultancy company in Finland. The company has a long history of and is experienced on both agile methods as well as producing information systems for the government. By the wish of the company, the client and the interviewees, all participants to the project shall remain anonymous.

A post-implementation group interview for the key personnel of the selected project was held. Semi-structured interview approach was used, where time was given to the interviewees to elaborate their thoughts about the phenomenon under study. The general questions concerned the scope and size of the project, amount of the personnel involved, and the daily routines of the team.

Additionally, the security standards that were applied to the project were gathered. The security mechanisms developed to implement the requirements were charted, along with how they were presented to the client and auditors. Finally, the amount of extra work caused by the security requirements was discussed and roughly estimated, and the interviewees recounted their views of the lessons learned in the project. The interview session also acted as a retrospective for the whole project, where the participants were able to express their views of positive and negative aspects of the project and the effect the security requirements had. The results of the interview were then analyzed by the researchers and the key observations were emphasized.

There were two interviewees: the Scrum Master and the head architect of the project; the latter was also responsible for the design of the technical stack, including the security features. In practice, the architect assumed a new role in the implementation team dubbed *security developer*. Both of the interviewees were essential to the project's implementation, and the one most capable of providing insight to the project background and its execution, as well as to evaluate its results and level of success. The interviewees were also the only team members that consistently participated in all of the sprints in the project, and were involved in the project for its whole duration.

The questions posed before the interviewees were divided into three groups. First three questions concerned the project background (Q1-Q3); following five questions concentrated on the project process, security standards, and feedback on the Scrum and security (Q4-Q8); and the final two questions canvassed the interviewees' views on the project results and success factors (Q9-Q10).

The questions asked in the interview were as follows:

Q1: Project subject and scope?
Q2: Project resources, budget, and duration?
Q3: Personnel locations, multi-site teams?

Q4: What VAHTI standards were followed?
Q5: What other security standards and regulation were included?
Q6: Other restrictions (safety, privacy, agency specific regulations)?
Q7: What *types* of steps were taken to enforce them?
Q8: How was the security assurance verified (audited) and audit trail maintained?
Q9: Did the budget and schedule hold, and what was the amount of extra work caused by security?
Q10: What were the lessons learned?

After the interview, some complementary questions were asked via emails to confirm certain details, but otherwise the initial interview session was deemed sufficient for the purpose of this study. Access to exact budget or workload figures, or system logs or other technical documentation was not made available for research: the security classification of the platform prevented using this data even for verification. Instead, the interviewees relied on their personal experience and notes made during the project, and provided best estimates on the matters in a general level accepted for publication.

5. Case description

The client agency required a VAHTI compliant IDM platform for their information systems, and for users and system administration and management purposes. The platform was to be built using off-the-shelf components, installed on common open source operating systems, and deployed onto a large scalable array of virtual servers. A similar IDM platform was built also to authenticate and manage the identities of the administrators who manage other VAHTI compliant servers and services, and is to be separately instantiated for regular office users as well based on the experience and solutions gained in this project.

The IDM was deemed a critical service in respect of agency's security, privacy and business requirements. Whereas the agency had 650 internal users connecting to 450 separate server-side computer systems, they also manage a sizable array of contractors with up to 12,000 users. The building project was conducted at the same time the server platform itself was being built, which added to the challenge in such way that all the requirements of VAHTI were to be met by a novel implementation.

Nearly all the design and definition work was to be completed in this project. To add to the challenge, the work was to be performed using Scrum, mainly to ensure steering group's visibility to the project's progress, and to enable reacting to any unexpected obstacles or hindrances met during the project execution. Unfortunately, for the project team, the client also saw use of Scrum as a method to change the project's scope during its execution by adding items to the product backlog, or removing them from there, which caused certain degree of confusion among the team and forced it to abandon some work already completed. These aspects of Scrum projects, however, are not a security issue but of a more generic field of project management, and therefore are not further discussed.

The development work consists of distinct phases, which were completed during one or more iterations:

1. *Definition*: synthesis of the requirements, component candidate selection, risk assessment and analysis.
2. *Design*: architecture design, definition of interfaces, component hardening plans.
3. *Development*: component research, modification (especially hardening the operating systems and software), and installation.
4. *Testing, reviews, audits and acceptance*: security testing, external audits and formal acceptance of the product to be a part of the agency's system portfolio. In effect, security assurance processes.

As there were no formal milestones preset at the beginning of the project, the security gates, such as audits, were passed flexibly whenever each feature was considered mature enough. This removed certain amount of unnecessary overhead, as a traditional fixed milestone dates may call for the team to work overtime, which may get costly due to pay compensations and cause delays to other projects due to resource shortage.

5.1. Project organization

The project involved an average of nine persons at any given time: Scrum Master, dedicated Product Owner, Security Architect (who, during sprints, was completing tasks in the role of a developer), and the developers split into their production teams based on location and occupation.

The service provider in charge of the project is a devout follower of ITIL[5], a well-established and recognized set of industry standard best practices for IT service management. As is typical for an ITIL-oriented organization, the infrastructure production teams reside in their respective "silos", with very little communication with other teams. Production teams were divided by their specialization. The platform teams involved in the project were "Storage and Backup", "Server Hardware", "Windows Operating Systems", "Linux Operating Systems", "UNIX Operating Systems", "Databases" and "Networks". The IDM application specialists came from their own team, a separate unit within the corporation.

This Scrum project brought together the specialists from these various teams at least for the daily 15-minute stand-up meeting – albeit most of the time virtually. Due to teams' multiple physically separated locations, the meetings were without exception held as telephone conferences.

The developers participating to the project in its different phases were so diverse that only the Scrum Master, security developer (i.e., the architect) and

[5] http://www.itil.org.uk/

the Product Owner participated in each sprint throughout the project. The developers were part of a larger resource pool and drawn into the sprints or spikes in various phases of the project whenever their expertise was required.

Much of the work related to VAHTI regulations was done in the planning phase: it turned out that in addition to VAHTI, the client agency had compiled their own list of requirements, which was based on VAHTI but had new security elements added to the public requirements. The client viewed this to be necessary to compensate the dropping the specific requirements for VAHTI compliant application development (VAHTI, 2013:1) in the beginning of the project.

The project extended over a period of 12 months, from planning phase to accepted delivery of final sprint. The amount of work was measured in story points, and the average velocity of each sprint was 43 points. Divided with the average number of the developers (9) and the length of the sprint (15 workdays) gives a rough estimate of a story point equaling three workdays. As an overall measure, the story points give an impression of the size of the tasks. This sort of conversion may not be meaningful in general and outside of the scope of a single project, as the story points are primarily used to compare the features (or stories) to each other within a single project. For purposes of this study, the fact that largest single units of security work, the hardenings, were not performed in sprints and therefore not measured in story points, makes pinpointing the cost of security work much harder. In this case, the interviewees' estimates were the only source of the amount of workload, and although trusted to be reliable, exact figures would have been preferred.

5.2. Project execution

From the beginning, the team's approach to the security tasks was pragmatic, although in terms of Scrum, rudimentary: stories that were found difficult to time-box at the time of their implementation were taken out of the sprint cycle and completed as spikes. Prime examples of such tasks were operating system hardenings, a task essential for the platform security: the project team allocated resources to these tasks, and just ran them as long as the tasks took. This resulted in a project structure presented in Figure 2, where there were major sidetracks to the main sprint cycle. As tasks such as these were in the very core of the project goals, it would have been beneficial to go through the trouble or even adjust the Scrum structure to better accommodate these items.

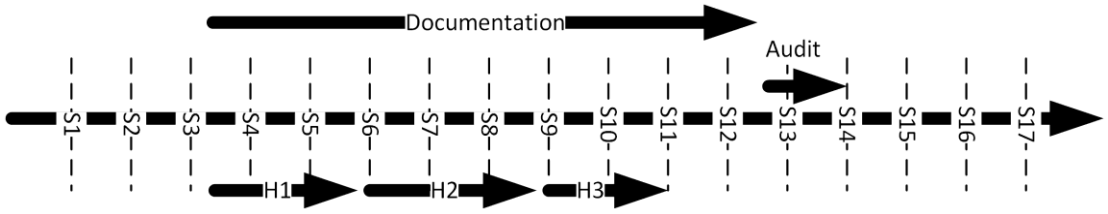

Figure 2. Project structure and spikes.

The sprints are represented as the main story line. The parallel lines represent the spikes that were executed outside the main sprint structure. Their results (deliverables) were demonstrated at a sprint demo after the spike had run its course, although they were executed independently without time-boxing. There were three distinct task types outside the sprint structure:

1. *System hardenings*, performed for each tier or environment of the system under development: Development, Quality Assurance (QA), and Production environments. The results obtained in the Development phase were not directly usable for the upper environments, whereas the QA environment was built to be production-like. As a result, the work done at QA phase was partly reusable at Production phase. Despite the technical similarities, the ITIL-guided maintenance models of these two environments were so great that the team proceeded in executing the Production environment hardenings as a spike as well.
2. *Documentation* was a ubiquitous process during the development. This included risk management, technical architecture and technical component documentation, test plans and reports. Documentation comprised most of the security assurance. Complete list of VAHTI requirements for documentation are presented in Appendix 3 of the VAHTI instruction 3/2012[6]. In this document, there are 224 mandatory requirements listed for the increased security level information systems. Almost all of these requirements call for some type of written evidence to be verified and reviewed, although most of the documentation artefacts are created in other than the development phase of the information system's life cycle.
3. *Reviews and audit* were performed based on the documentation and included physical testing of implementation.

The demand for increased security (literally, the "increased level" on VAHTI security classification) also stated how the systems were deployed: to maintain audit trail, all changes to the production environment, including all server and hardware installations during its buildup, were performed following ITIL processes. These processes added extra levels of bureaucracy, and the team reported getting acceptance from the Change Advisory Board (CAB) for all changes to be made in the production environment had a very adverse effect on the deployment schedules. Combined with the policy of role separation between developers and maintenance personnel, this caused the building and installation of the production environment to be document-driven, bureaucratic and slow. The policy of separating the roles of developers and maintenance effectively prevents the DevOps type of continuous delivery maintenance model, and would require e.g. a form of "continuous security" model, such as presented by Fitzgerald & Stol (2014).

In this project, the continuous delivery model was used with the lower environments, speeding the rate of delivery significantly. When building the production environment, the flow of work assumed in previous sprints was

6 https://www.vahtiohje.fi/web/guest/708 (available in Finnish only)

disrupted, which caused unnecessary slowness and cost overhead. Documentation necessary for the maintenance personnel was to be created before the handover, and as such did not necessarily contain all the required information and details. Mandatory use of ITIL processes when building the production environment was one of the main schedule hindrances of the project according to the interviewees.

Depending on the items in the current sprint backlog, the team was divided in two or three geographically separated locations during the whole length of the project. The organizational separation of the developers resulted in situation, where even the persons based on the same location did not necessarily sit near each other or communicate with other team members directly. The central location for the project, and the physical location of the server platform was Helsinki, Finland, but the team members were divided on several sites. The Scrum Master performed most of her duties remotely, without being in direct contact with the developers except rarely. As usual in large ICT service companies, almost all developers were also involved in other projects at the same time. The overall experience of the team was deemed very high, although in infrastructure work the use of agile methods is not very common, and is customer dependent at best. As per this fact, most personnel was mostly inexperienced with Scrum, although they received basic Scrum training before and during the project. Use of Scrum was reflected by the use of collaboration and project management tools, most importantly Atlassian JIRA[7] specifically customized for the agency's use. The Scrum Master promoted and demanded the use of JIRA as reflecting the work performed in daily sprints. The Product Owner's most visible role was following the project's progress based on what team members reported on this tool. In general, the team was reported to be happy or at least content with Scrum, at least up until the production environment-building phase where ITIL processes broke the team's workflow.

The requirements called primarily for well-documented software quality and component and process security. Most of the additional work was directly security related, and creating its documentation. The platform also had strict and formal requirements for availability and reliability. Outside the security domain, the main source of regulation-related work was duplication of all infrastructure into the service provider's second data center. The data centers themselves, as well as the personnel administering the system and its infrastructure were subject to meticulous security screening. Proper level of access control was enforced, the server rooms' CCTV system extended to cover the new servers, and remote connection practices were reviewed. All personnel involved with the client was to be security checked by the national Finnish Security Intelligence Service[8]. Data itself must reside within the country's borders and even the infrastructure's configuration data and work tickets in the Configuration Management Database (CMDB) were to be made inaccessible for personnel who are not security checked.

---

7 https://www.atlassian.com/software/jira/agile

8 http://www.supo.fi/security_clearances

As an infrastructure project, the main technical obstacle was securing the hardware, operating systems, middleware and the application (the IDM system) against security threats. The bulk of this work was performed by one of the interviewees, the security developer. Hardening in this case covered analyzing and removal, or blocking, of hardware and software features, and testing against the threats. The purpose is to reduce the attack surface of the platform under construction and protect it from both internal and external threats, as well as minimize the components where potential future vulnerabilities may emerge.

On hardware level, hardening means controlling the network interfaces and the surrounding local area network, routing and traffic rules. It also covers all hardware maintenance interfaces, typically accessible through the network. On operating system and software level, the operating system's or software manufacturers, such as Microsoft, provide their own hardening instructions, which were used as a baseline. These were combined with the best practices of the consultant company's own experiences and policies, and the explicit instructions and requirements given by the client organization. These included uninstalling a large number of modules and services, disabling a number user accounts and policies, and enforcing a number of others, and restricting access and privileges throughout the system. The same principles were applied to each software component installed on the server platform.

By definition, all access rules and user validations had to be applied to the infrastructure services provided for the server platform; these include software and hardware patching, network access, malware protection, hardware and application monitoring, and backups. The inherent uncertainty of security testing, together with the inter-dependency of the components affected by the removal and alteration of the services and restriction of rights made predictable time-boxing of these tasks so unreliable that the team decided to execute them as spikes.

5.3. Cost of security work

The Scrum Master estimated that the extra work caused by the regulations was approximately **25 to 50% of the project's total workload** and, in practice, the duration of the project. As accurate billing information was not made available for the researchers, this was accepted as the best estimate of the real cost of the security work. Most of the overhead comprises from the documentation of the solutions. Security-related documentation was created by all team members: project manager and the security developer (architect) created most of the documentation, and the Product Owner as the client's representative made sure that the correct regulations were applied.

Developers were burdened by creating appropriate level of security-oriented technical documentation of all their work, especially related to operating system and application hardening procedures. The hardening process itself lasted for four months, presenting the largest tasks in the project. Changes to the

production environment were further complicated by ITIL's requirement of strict Change Advisory Board processing of each change that was made.

6. Analysis

The research objective for this study is to identify best practices as well as hindrances of using agile software development. This case provides a good view how unmodified Scrum lent itself to a situation, where a large amount of regulations caused extra work with uncertainties in work estimates. Due to these uncertainties, or the large amount of presumably indivisible work included in some of these tasks, the team was simply not able to fit certain features into the sprint structure. Additionally, in contradiction to traditional security view, iterative and incremental approach to development and building forced the project team, steering group and the client to rethink how the product's and its management's security assurance was to be provided. In a sequential waterfall model, the security deliverables and tasks were tied into the predetermined milestones, without the flexibility provided by Scrum. As presented in Figure 2, the project was in practice executed partly following a "waterfall" model, yet without milestones fixed in advance; these waterfall processes ran alongside the main project, and their deliverables were then included in the project outcomes.

Based on the above, in the strictest sense the project organization failed utilizing Scrum methodology to create the product, although the superficial requirements were fulfilled – the client was mostly interested in progress reports and the timely delivery of the complete and standard compliant product. The failures were partly due to inflexibilities on both the company developing the system, and the client demanding a formal and fixed approach to Scrum. Sprint planning for tasks, for example, called for features to be completed during the sprint. When this was already known to be extremely unlikely, these features were agreed to be performed as spikes. In retrospect, this was most likely caused by the thinking that security features were perceived as *overhead* and not actual features in the product, while in reality the security features were essential to the product itself. The resulting implementation model is partially waterfall-like.

Even without applying any formal modifications to Scrum, at least one of the "secure Scrum" features, presented in Section 3 and
Figure 1, was taken into use, as the project architect assumed the role of security developer. In practice, most of the physical work triggered by security requirements was done in spikes outside the sprints. When the work is done in a non-iterative way, just letting them run along the project, the benefits of Scrum are lost. Based on the project manager's estimate of cost increase factor was 1.5-2x, caused by the security features, and thus there exists a large saving potential in rearranging the security work. Attempting a new approach and restructuring the work into iterations is recommendable in future projects. Initial spikes are acceptable, but in this case, the team failed to utilize the experience gained from them, and continued to implement similar security features as spikes even after the first one. This is represented in Figure 2 by the OS hardening spikes H1, H2

and H3. During the spikes, there was very little activity in the actual sprints, as also documentation was done as a spike.

The team defended their selected approach by stressing the inherent differences in the physical environment and management practices of the development, quality assurance and production environments, but also from the undertones of the developer's interview, it was perceivable that the attitude towards using Scrum in this kind of project was negative to start with. Time-boxing the uncertain tasks to three-week sprints, having to perform the demonstrations after each sprint, and other Scrum routines were perceived to some degree as distractions from the main work. This mentality seemed to affect some members of the team despite the personnel was trained in the Scrum method and the tools necessary.

During the interview, the team was uniform on the key success factors of the project. They emphasized the importance of document management, and very strict requirement management. The amount of overlapping and sometimes outright conflicting security requirements even within the VAHTI requirements increased the Scrum Master's workload substantially. Use of Scrum was deemed to have overwhelmingly positive effect, by enabling faster reaction to changes in the requirements and directness of the client feedback. In addition, the team praised frequent sprint planning for the effect of keeping the team focused, especially in contrast to the very long spikes run during the project. In retrospect, the team regretted not utilizing the Product Owner more already in the beginning, as direct channels to the client were viewed to be very valuable during the implementation. Furthermore, the client's key personnel were not always present at sprint demos, which caused unnecessary questions and insecurity on the client's side, despite the features were already completed and already once comprehensively demonstrated.

The effect of Scrum to the efficiency of the work was estimated very positive. The extra cost of the security was partly compensated by the fact that rigorous testing and documentation of the technical solutions had also a positive impact on the quality of the work, improving the system's reliability and availability. It can also be argued that the cost of security work is lower when it is done proactively rather than repairing an old system or trying to recover a breached one.

7. Discussion

There are three key findings in this study:

- First, an agile development method works in a security-regulated environment. This study showed that is possible to develop a system with set governmental security regulations by utilizing an agile method (Scrum). While the evidence is based on a single case, when combined with other evidence it shows that the oft-repeated belief of agile methods

being unsuitable for security engineering (cf. Rindell, Hyrynsalmi, and Leppänen 2017) seems not to hold.

- Second, Scrum as a method appears highly applicable for the software security engineering projects. In this case, only little modifications were needed to the method for meeting the security regulation restrictions. Yet, the team constantly faced "surprises", and were forced to adopt new techniques and models in order to avoid pitfalls. While this is not a hoped approach, the adoption to surprising changes is in the hearth of agile software development methods and the Scum's empirical software process improvement principle.
- Third, the interviewees reported that required security routines and their documentation took up to 25-50 % of the project's total budget. The time reports were not made available for researchers, yet this was the estimate given by the project manager.

This study has presented a case of building an infrastructure and setting up an identity management software platform for a governmental client. The client agency had a definitive set of security regulation and requirements: the VAHTI instructions. In addition to the government requirements, the service provider contracted to build the system was committed to several international ISO/IEC standards, as well as to their own management frameworks. Additionally, the project management was burdened with complex financial reporting tools and rules. Both the agency and the service provider's project management offices required employing the Scrum methodology as the project management framework. The research was conducted as post-project semi-structural interviews, and the information was gathered based on interviewees' experiences and personal notes of the project. The parties involved are anonymized, and only publicly available information about the project and the regulations involved was to be disclosed.

Scrum was initially applied in its standard form, with no formal security extensions. Security engineering activities were integrated into the product backlog, and performed within sprints whenever possible. During the project, the team adapted to the security work by creating a *de facto* security developer role, and many of the security engineering tasks ended to be performed outside of the regular sprint structure. Typically, security assurance is based on evidence gained through security testing, which also in this case had an adverse effect on the team's ability to schedule and time-box the items that were subject to these tests; these were performed as spikes instead. The same technique was also applied to documentation, which was performed outside the main sprints, and audits and reviews, which were separately scheduled one-time tasks. The results of these spikes were still presented in sprint demos among the other artefacts and results. The reported issues at product deployment in production environment prompt for developing and applying a delivery model that provides the required security assurance without the interruption to iterative development.

The team viewed the use of Scrum as a positive factor to project cost and quality, although arguably Scrum was not utilized to the maximum extent: important

parts of the work were done in spikes outside of the main sprint flow, without attempts to utilize the experience gained from them to time-box the future tasks. This was seen to benefit the project, although an iterative and more exploratory approach to those tasks might have proved more benefits in the long term, and it is still a possibility that the experience gained in this project can be utilized in similar future projects. The project team still regarded the security engineering activities and providing the required security assurance to compose a significant amount of extra work: at final stages, the workload effectively doubled. The initial approach in this project was more or less an unmodified textbook example of the Scrum method, but the team applied naturally certain security extensions. Conducting weekly **product backlog refinement sessions** was deemed essential for the project's success.

This project was a model case of two large entities that have decided to fit their organizations to work according to an agile framework. The nature of work itself has not changed, although the introduction of growing amount of security engineering and increasing regulation put an additional strain on the project's requirement management. Agile methods have inherent preference to produce working solutions instead of spending time documenting them; in contradiction to this goal, the documentation of the solutions is a key deliverable in the field of security. Scrum will continue to be used by both organizations. As the team's experience grows, we also expect the cost of the secure systems development to drop, while their quality and security gets better.

Based on the experiences gained in this case, Scrum has shown the potential to be suitable for security-oriented development work. With certain additions and modifications, it can be used to provide the security assurance required by the regulators in the ICT and software industry. Especially when applied by an organization capable to adjust itself to fully utilize the flexibility of incremental agile frameworks, instead of partially reverting to sequential mode of operations. We are yet to observe a pure agile project where security standards are in a central role: truly integrating security engineering processes and security assurance activities without losing the agile values and benefits gained by the use of those methods is still a work in progress.

Naturally, this study has its limitations. First, the analysis is based on a single case and overgeneralization of the results should be avoided. As study is by its design explorative, restricting to a single case is understandable. However, further work are needed to verify the results with new cases. Furthermore, case studies should be extended to cover also other agile methodologies used in software security engineering than Scrum. While Scrum is among the most popular development methods nowadays, it still present only a handful of different methods, tools and techniques developed inside the agile community.

Second, due to the nature of the project, non-disclosure agreements and security classifications, the researchers could not access the project documentation and verify the project team’s interpretations. Thus, no proper data triangulation with written documentation could have been done.

Third, the study presents a case where software security engineering succeeded well with a selected agile method. However, as the project faced only small disturbances that were able overcome with simple modifications, complete view on the methods suitability for complex security project cannot be assessed. A comparative study with, e.g., student teams handling a complex security project with agile, as well as traditional method will reveal more insights into this issue.

Finally, this study opens further avenues for research. Our study reported that the development team and manager estimated that almost half of the project's budget was spent on the security related tasks. Regardless of the exact amount, this finding calls for further development work on revealing the real cost of security as well as methods and tools to reduce time spent on security issues.

In addition, while Scrum was shown to cope with security development, it is clearly not perfect fit for the work. Thus, future work should be focused on developing, testing and validating tools, techniques and models to extend Scrum or other methods to be more suitable for security development projects.

8. Conclusion

This chapter presented an exploratory case study on a security development project regarding a governmental information system with strictly regulations. The aim was to explore whether agile was a successful approach for the development work or not. The result shows that agile development, performed using the Scrum method, is suitable also for security engineering work. While drawing too far-reaching conclusions from a single case study would be ill advised, this case still clearly contradicts the criticism against agile methods' suitability for security engineering. Among the key factors to success were Scum's iterative approach, enhancing the management of client's strict security requirements. Use of Scrum processes, artifacts and roles also improved communication both within the team and towards the client.

In contrast, the observations also reveal certain negative issues in the Scrum method, and the way agile values and principles are affecting the security development. The findings of this study suggest the requirement for new tools, techniques and models to solve the challenges and alleviate the issues in agile software security engineering. The solutions include security training for all the project participants, improved mechanisms to manage security requirements, and techniques to the security tasks into iterative process.

Acknowledgments

The authors gratefully acknowledge TEKES, the Finnish Funding Agency for Innovation, DIMECC Oy, and the Cyber Trust research program for their support.

References

Baca, D. & Carlsson, B. (2011). Agile development with security engineering activities. Proceedings of the 2011 International Conference on Software and Systems Process, ICSSP '11 (pp. 149-158). ACM.

Beznosov, K., & Kruchten, P. (2004). Towards agile security assurance. NSPW '04 Proceedings of the 2004 workshop on New security paradigms (pp. 47-54).

Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., and Kruchten, P. (2006). Extending XP Practices to Support Security Requirements Engineering. In Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems (pp.11-18). New York, NY, USA, ACM.

Chivers, H., Paige, R., and Ge, X. (2005). Agile security using an incremental security architecture. Proceedings of the 6th international conference on Extreme Programming and Agile Processes in Software Engineering, June 18-23, 2005, Sheffield, UK.

Creswell, J.W. (2003). Research Design: Qualitative and Quantitative and Mixed Methods Approaches. SAGE Publications, Inc., Thousand Oaks, California, 2nd edition.

Cruzes, D. S., Felderer, M., Oyetoyan, T. D., Gander, M., & Pekaric, I. (2017). How is security testing done in agile teams? a cross-case analysis of four software teams. In Baumeister, H., Lichter, H., and Riebisch, M., editors, Agile Processes in Software Engineering and Extreme Programming, pages 201–216, Cham. Springer International Publishing.

Diaz, J., Garbajosa, J., & Calvo-Manzano, J.A. (2009). Mapping CMMI Level 2 to Scrum Practices: An Experience Report. Software Process Improvement, volume 42 of Comm. in Computer and Information Science (pp 93-104).

Fitzgerald, B. & Stol, K-J. (2014). Continuous software engineering and beyond: Trends and challenges. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, RCoSE 2014 (pp. 1-9), New York, NY, USA. ACM.

Fitzgerald, B., Stol, K-J., O'Sullivan, R., & O'Brien, D. (2013). Scaling agile methods to regulated environments: An industry case study. Proc. of International Conference on Software Engineering, ICSE '13 (pp 863-872).

Ge, X., Paige, R. F., Polack, F. & Brooke, P. (2007). Extreme programming security practices. Agile Processes in Software Engineering and
Extreme Programming, (pp. 226-230), LNCS 4536.

Heeager, L. & Nielsen, P. (2018). A conceptual model of agile software development in a safety-critical context: A systematic literature review. Information and Software Technology, Volume 103, 2018, Pages 22-39, ISSN 0950-5849.

Hope, P., McGraw, G. & Anton, A. I. (2004). Misuse and abuse cases: getting past the positive. IEEE Security & Privacy, Volume 2, Issue 3, pp. 90-92.

ISO/IEC, (2008). Information Technology - Security Techniques - Systems Security Engineering - Capability Maturity Model (SSE-CMM) ISO/IEC 21817:2008.

ISO/IEC, (2013). Information Technology - Security Techniques - Code of Practice for Information Security Controls. ISO/IEC 27002:2013.

Kasauli, R., Knauss, E., Kanagwa, B., Nilsson, A. & Calikli, G. (2018). Safety-Critical Systems and Agile Development: A Mapping Study. Accepted at Euromicro Conf. on Software Engineering and Advanced Applications 2018, Prague, Czech Republic.

Kongsli, V. (2006). Towards agile security in web applications. Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications, October 22-26, 2006, Portland, Oregon, USA

Morrison, P., Smith, B,. & Williams, L. (2017). Surveying Security Practice Adherence in Software Development. In Proceedings of the Hot Topics in Science of Security: Symposium and Bootcamp (HoTSoS). ACM, New York, NY, USA, 85-94. DOI: https://doi.org/10.1145/3055305.3055312

b. Othmane, L., Angin, P., Weffers, H., and Bhargava, B. (2014). Extending the Agile Development Process to Develop Acceptably Secure Software. IEEE Transactions on Dependable and Secure Computing 11(6): 497--509.

Pietikäinen, P. & Röning, J. (2014). Handbook of the Secure Agile Software Development Life Cycle. Univ. of Oulu.

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. Information Systems Journal Vol. 20, No. 5 (pp. 449–480). https://doi.org/10.1111/j.1365-2575.2007.
00259.x

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2015:1) A comparison of security assurance support of agile software development methods. Proceedings of Proceedings of the 15th International Conference on Computer Systems and Technologies (pp. 61-68). ACM.

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2015:2) Securing Scrum for VAHTI. CEUR Workshop Proceedings (pp. 236-250) Vol. 1525.

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2017) Busting a Myth: Review of Agile Security Engineering Methods. Proceedings of the 12th International Conference on Availability, Reliability and Security (pp. 74:1-74:10). ACM.

Sonia, Singhal, A., Banati, H. (2014) FISA-XP: an agile-based integration of security activities with extreme programming, ACM SIGSOFT Software Engineering Notes, 39(3):1-14.

VAHTI (2009:2) ICT-toiminnan varautuminen häiriö- ja erityistilanteisiin. URL https://www.vahtiohje.fi/web/guest/2/2009-ict-toiminnan-varautuminen-hairio-ja-erityistilanteisiin. Retrieved 15/08/2017.

VAHTI (2012:2b) Requirements for ICT Contingency Planning. URL https://www.vahtiohje.fi/web/guest/2b/2012-requirements-for-ict-contingency-planning. Retrieved 15/08/2017.

VAHTI (2013:1) Sovelluskehityksen tietoturvaohje. URL https://www.vahtiohje.fi/web/guest/vahti-1/2013-sovelluskehityksen-tietoturvaohje. Retrieved 15/08/2017.

VAHTI (2012:3) Teknisen ympäristön tietoturvataso-ohje. URL https://www.vahtiohje.fi/web/guest/3/2012-teknisen-ymparistontietoturvataso-ohje. Retrieved 15/08/2017.

VersionOne, (2017). 11th annual state of agile survey. https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 Retrieved 15/08/2017.

Villamizar,H., Kalinowski,M., Viana, M., & Méndez Fernández, D. (2018). A Systematic Mapping Study on Security in Agile Requirements Engineering. Published at the Euromicro Conference on Software Engineering and Advanced Applications 2018.

Wäyrynen, J., Bodén, M., and Boström, G. (2004). Security Engineering and eXtreme Programming: An Impossible Marriage? Springer Berlin Heidelberg, Berlin, Heidelberg, (pp. 117—128).

Yin, R. K. (2003). Case Study Research: Design and Methods. SAGE Publications, Inc., 3rd edition.