International Journal of Secure Software Engineering
Volume 8 • Issue 1 • January-March 2017

Case Study of Agile Security Engineering: Building Identity Management for a Government Agency

Kalle Rindell, Informaatioteknologian laitos, University of Turku, Turku, Finland
Sami Hyrynsalmi, Tampere University of Technology, Pori, Finland
Ville Leppänen, Informaatioteknologian laitos, University of Turku, Turku, Finland

ABSTRACT

Security concerns are increasingly guiding both the design and processes of software-intensive product development. In certain environments, the development of the product requires special security arrangements for development processes, product release, maintenance and hosting, and specific security-oriented processes and governance. Integrating the security engineering processes into agile development methods can have the effect of mitigating the agile methods' intended benefits. This article describes a case of a large ICT service provider building a secure identity management system for a sizable government agency. The project was a subject to strict security regulations due to the end product's critical role. The project was a multi-team, multi-site, standard-regulated security engineering and development work executed following the Scrum framework. The study reports the difficulties in combining security engineering with agile development, provides propositions to enhance Scrum for security engineering activities. Also, an evaluation of the effects of the security work on project cost presented.

KEYWORDS

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

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[1]. 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. Alnatheer, Gravel & Argles, 2010; Baca

DOI: 10.4018/IJSSE.2017010103

Copyright © 2017, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

& Carlsson, 2011; Beznosov & Kruchten, 2004; Fitzgerald, Stol & Sullivan, 2013; Ge, Paige, Polack & Brooke, 2007; Pietikäinen & Röning, 2014; Rindell, Hyrynsalmi & Leppänen, 2015), 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 best practices as well as hindrances 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 described, a Scrum project was conducted with the objective of building an IDM system for VAHTI-compliant information systems, and a secure VAHTI-compliant server platform to host the systems, including the IDM. The server platform was to be used also to host software development projects (with certain dispensations). The project was executed during 2014 and 2015, and had a duration of 12 months. The development team was split into two to three geographically dispersed groups, with the actual amount of teams involved dependent on the tasks at hand and the overall phase of the project. As a standing practice with the government agency that initiated the building of the platform, the project was managed using 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 steering group. 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 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)[2]. 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, and how the project was conducted. As the study revealed that not all the objectives of using `pure' Scrum were not met, suggestions are made to improve the efficiency of the development work by introducing rudimentary security engineering extensions to the Scrum framework. The modifications include a new role for a security developer, and also suggest specific security sprints and other security-oriented additions to the run-of-the-mill Scrum. We also discuss how the introduction of the security engineering activities into the project affect cost, efficiency and the conduct of the project.

2. BACKGROUND AND MOTIVATION

The use of agile methods has become an industry practice, 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. Based on the literature, and also the findings this observed case, the typical approach to agile security engineering is to simply start using the methodology at hand without formal adjustments, with the notable exception of thorough and formal approach to security engineering described 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. In this approach, the security-related tasks are treated simply as items in the backlog: the security requirement items are converted to tasks, given story points, and completed among the other items as best seen fit. When security items which cannot reasonably be time-boxed because of the inherent uncertainties of the work, or the inexperience of the team, they separated from the Scrum sprint cycle and completed in non-time-boxed spikes.

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 provide the best security assurance for the end product. Achieving security assurance is by all means possible with careful planning, although lacking in 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 and 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 security engineering tasks, the security cost overhead can be reduced while the security of the end product is enhanced, when 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 end products.

The next subsections provide more information about VAHTI, and the use of Scrum methodology in development projects requiring security standards compliance. Due to similarities in the requirements, the same observations and recommendations we make in this paper are found applicable also to software safety regulations, in e.g. medical field.

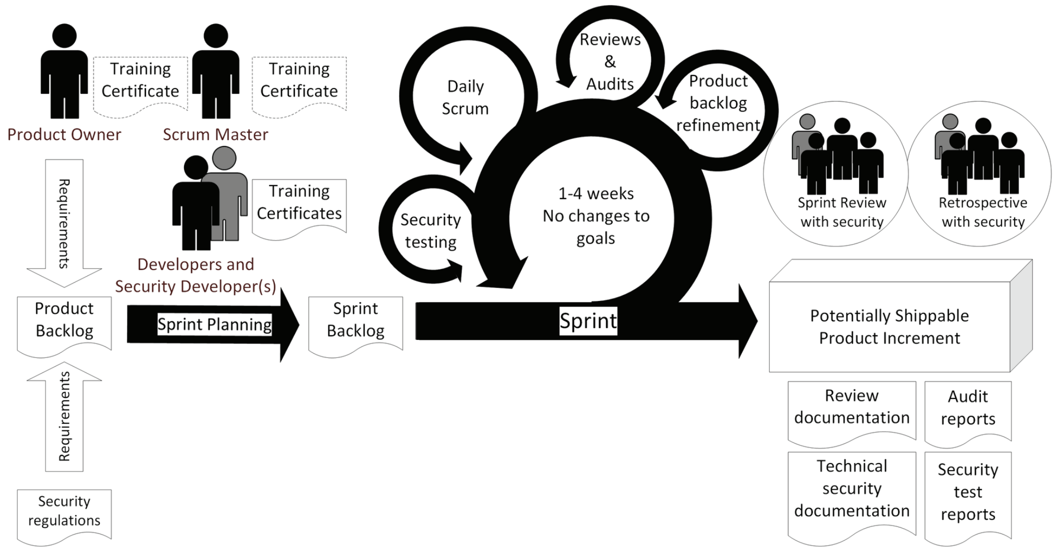
2.1. Security-Augmented Scrum

Scrum is a generic 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.

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 artifacts, 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

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



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.

2.2. Security Regulations and Standards Applied

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", FMoF (2009)
- VAHTI 2b/2012 "Requirements for ICT Contingency Planning", FMoF (2012)
- VAHTI 3/2012 "Instructions for Technical Environment Security", FMoF (2012)

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 article. 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. As a curious example, the Finnish name of VAHTI board itself has changed recently, albeit the English translation has not.

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 reporting responsibilities.

2.3. Data Security in VAHTI

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 I to IV, where IV is 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 was encrypted despite the official classification level.

2.4. About Security Engineering

The term `security engineering' in software industry comprises of all security-related tasks within a software-intensive product's life cycle. The standard's way to categorize these activities is to divide them into three main process areas: risk, engineering and assurance (see ISO/IEC 21817). The risk process assesses the risk and aims in minimizing it by assessing threats and vulnerabilities, and the impact they have, producing risk information. The security engineering process uses this information with other security-related input to define security needs and provides solutions to fill them. 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 risk, and define the impact and actions to be taken when the residual or unrecognized risk is realized: what will happen when things break.

3. RESEARCH PROCESS

This study follows a case study design method by Yin (2003), and a qualitative research approach by Cresswell (2003). For the study, we were looking for a development project that was both using agile methods and fulfilling VAHTI regulations. Our decision was to focus on the VAHTI regulations, as they are viewed to be a national standard and, therefore the number of possible cases would be higher. In addition, we were looking for a project which would be either ended or near its ending in order that we would be able to evaluate the success of the used model. Finally, the selected case should be a representative candidate as well as be able to produce rich information about the phenomenon under study.

We ended up to select a project case where identity management and verification service was ordered by a governmental customer who required the use of VAHTI. The development work 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 should be representative.

The project was executed by a mature, well-known software product development and consultancy company in Finland. The company has a long history of 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.

For this study, we held a post-implementation group interview for the key personnel of the selected project. We used a semi-structured interview approach 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.

Also, 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.

The project was selected as a potential research target due to its strict security requirement and the fact that it was executed and managed using Scrum framework. The interviewees were the Scrum Master and the head architect of the project. They were both deemed as key personnel of the project, and they were able to provide insight to the project background, its execution as well as its results. The selected interviewees were also the only ones that persistently participated in all of the sprints 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 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.

4. CASE STUDY: THE PROJECT

The agency required a VAHTI compliant IDM platform for their various 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 to be a critical service in respect of agency's security, privacy and business requirements: while the agency had 650 internal users connecting to 450 separate server-side computer systems, they also manage a sizable array of contractors with a total user amount of up to 12,000. 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 also to enable reacting to any unexpected obstacles or hindrances met during the project execution. Unfortunately for the project team, the customer 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 (i.e., hardening), and installation.
4. **Testing, Reviews, Audits and Acceptance:** security testing, external audits and formal acceptance of the end 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 to be 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.

4.1. Project Organization

The project involved an average of nine persons at any given time: a Scrum Master, a dedicated Product Owner, a Security Architect (in basic scrum, part of the development team in the role of a developer), and the developers split into their production teams based on location and occupation.

The service provider is a devout follower of ITIL[3], a well-established and recognized set of industry standard best practices for IT service management. Typically for an ITIL-oriented organization, the infrastructure production teams reside in their own "silos", with very little communication with other teams. Production teams were divided by their specialization, in this case `Storage and Backup', `Server Hardware', `Windows Operating Systems', `Linux Operating Systems', `UNIX Operating Systems', `Databases' and `Networks'. In addition, the IDM application specialists came from their own team, residing within a separate unit within the company. The project brought specialists from these various teams together, at least virtually --- for at least the daily 15-minute stand-up meeting. Due to team's multiple physically separated locations, the meetings were without exception held as telephone conferences.

The teams were utilized in different phases of the project in such way that only the Scrum Master, security developer (i.e., the architect) and the Product Owner had personal activities in every single 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.

4.2. Project Execution

Much of the work related to VAHTI regulations was done in the planning phase: it turned out that 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. This partially to compensate the dropping the specific requirements for VAHTI compliant application development (FMoF, 2013) in the beginning of the project.

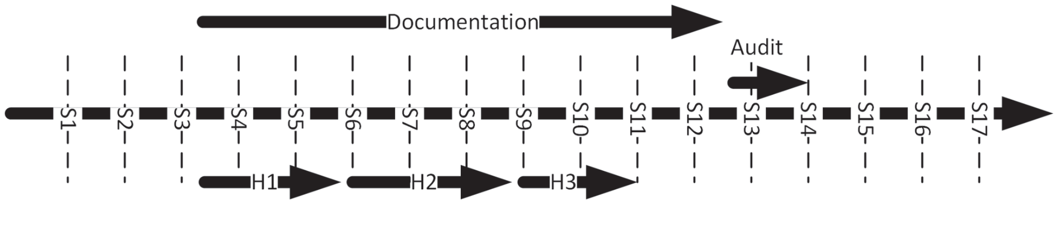
The project extended over a period of 12 months, from planning phase to accepted delivery of final sprint. The amount of work was measured in story points, and the average velocity of each sprint was 43 points. Divided with the average number of the developers (9) and the length of the sprint (15 work days) gives a rough estimate of a story point equaling three work days. 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.

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 side tracks 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.

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, 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

Figure 2. Project structure and spikes



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[4]. 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.

4.2.1. Product Deployment Model

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 disrupted, which caused unnecessary slowness and also 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.

4.2.2. Team Structure and Tools

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 in the vicinity of 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 were 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[5] 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.

4.2.3. Security Story Types and Their Implementation

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[6]. 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.

4.2.4. Technical Tasks: System Hardenings

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 any and 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.

4.3. 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 work load. As accurate billing information was not available, 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-based requirement of strict Change Advisory Board processing of each change that was made.

5. ANALYSIS AND DISCUSSION

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. Also, in contradiction to traditional security view, iterative and incremental approach to development and building forced the project team, steering group and also the client to rethink how the end 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 -- customer was mostly interested in progress reports and the timely delivery of the complete and standard compliant end 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.

Even without applying any formal modifications to Scrum, at least one of the "secure Scrum" features, presented in Chapter 2.1 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 is 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. 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 quite 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. Also the team praised the frequent sprint planning's effect of keeping the team focused, in comparison 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. Also, 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.

6. CONCLUSION AND FUTURE WORK

This study has presented a case of building an infrastructure and setting up an identity management software platform for a governmental customer. The customer agency had their own set of security regulation and requirements, namely 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 their own management frameworks and sometimes complex financial reporting 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 in a post-project semi-structural interview, and the information was gathered based on their experiences and 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 artifacts 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 work load 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. Simply 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, and as the team's experience grows, we expect also 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 back 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.

ACKNOWLEDGMENT

The authors gratefully acknowledge Tekes - the Finnish Funding Agency for Innovation, DIMECC Oy, and the Cyber Trust research program for their support. This paper extends a conference paper “Case Study of Security Development in an Agile Environment: Building Identity Management for a Government Agency” by Rindell, Hyrynsalmi & Leppänen (2016).

REFERENCES

Alnatheer, A., Gravell, A., & Argles, D. (2010). Agile security issues: A research study. *Proceedings of the 5th International Doctoral Symposium on Empirical Software Engineering (IDoESE).*

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. *Proceedings of the 2004 workshop on New security paradigms NSPW '04* (pp. 47-54).

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

Diaz, J., Garbajosa, J., & Calvo-Manzano, J. A. (2009). Mapping CMMI Level 2 to Scrum Practices: An Experience Report. In Software Process Improvement, CIS (Vol. 42, pp. 93-104).

Fitzgerald, B., & Stol, K.-J. (2014). Continuous software engineering and beyond: Trends and challenges. *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, RCoSE 2014* (pp. 1-9), New York, NY, USA. ACM. doi:10.1145/2593812.2593813

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). doi:10.1109/ICSE.2013.6606635

FMoF. (2009) ICT-toiminnan varautuminen häiriö- ja erityistilanteisiin. Retrieved from https://www.vahtiohje.fi/web/guest/2/2009-ict-toiminnan-varautuminen-hairio-ja-erityistilanteisiin

FMoF. (2012) Requirements for ICT Contingency Planning. Retrieved from https://www.vahtiohje.fi/web/guest/2b/2012-requirements-for-ict-contingency-planning

FMoF. (2012) Teknisen ympäristön tietoturvataso-ohje. Retrieved from https://www.vahtiohje.fi/web/guest/3/2012-teknisen-ympariston-tietoturvataso-ohje

FMoF. (2013) Sovelluskehityksen tietoturvaohje. Retrieved from https://www.vahtiohje.fi/web/guest/vahti-1/2013-sovelluskehityksen-tietoturvaohje

Ge, X., Paige, R. F., Polack, F., & Brooke, P. (2007). Extreme programming security practices. In Agile Processes in Software Engineering and Extreme Programming, LNCS (Vol. 4536, pp. 226-230). doi:10.1007/978-3-540-73101-6_42

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.

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

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2015). 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. doi:10.1145/2812428.2812431

Rindell, K., Hyrynsalmi, S., & Leppänen, V. (2015). Securing Scrum for VAHTI.*ProceedingsCEUR Workshop*(pp. 236-250).

VersionOne. (2016). 10th annual state of agile survey. Retrieved from https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf

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

ENDNOTES

1 https://www.vahtiohje.fi/web/guest/home
2 https://www.viestintavirasto.fi/en/cybersecurity/ficorasinformationsecurityservices/ncsa-fi.html
3 http://www.itil.org.uk/
4 https://www.vahtiohje.fi/web/guest/708 (available in Finnish only)
5 https://www.atlassian.com/software/jira/agile
6 http://www.supo.fi/security_clearances

Kalle Rindell is an enthusiast of computers, Internet and security, with working experience of nearly two decades as a programmer and R&D engineer. He is currently completing his PhD at University of Turku, Finland in the fields of security and agile software development, while working for the CGI Group as a consultant.

Sami Hyrynsalmi, DSc (tech), is a nerd who has always enjoyed working with programming and computers. After graduating as MSc in software engineering from the University of Turku in 2009, he decided to focus on the real issues and started his doctoral dissertation work on mobile application ecosystems. After successfully defending his thesis in 2014, he has focused on various themes from software and its production to business ecosystems, software metrics as well as to computer games. Currently, he is working as an Assistant Professor (tenure track) of Software Product Management and Business in TTY Pori at Tampere University of Technology.

Ville Leppänen is a professor in software engineering and software security at the University of Turku (UTU), Finland. He has 180 international conference and journal publications. His research interests are related broadly to software engineering and security, ranging from software engineering methodologies, project management practices, and tools to security and quality issues, and to programming languages, parallelism, and architectural design topics. Leppänen is currently leading six research and development projects. He acts as the head of Software Engineering (UTU) and leader of Software Development Laboratory of Turku Centre for Computer Science.