Case Study of Security Development in an Agile Environment: Building Identity Management for a Government Agency

Kalle Rindell, Sami Hyrynsalmi, Ville Leppänen
Department of Information Technology
University of Turku
Turku, Finland
{*kakrind, sthyry, ville.leppanen*}*@utu.fi*

***Abstract*—In contemporary software development projects and computing tasks, security concerns have an increasing effect, and sometimes even guide both the design and the project's processes. In certain environments, the demand for the security becomes the main driver of the development. In these cases, the development of the product requires special security arrangements for development and hosting, and specific security-oriented processes for governance. Compliance with these requirements using agile development methods may not only be a chance to improve the project efficiency, but can in some cases, such as in the case discussed in this paper, be an organizational requirement.**

This paper describes a case of building a secure identity management system and its management processes, in compliance with the Finnish government's VAHTI security instructions. The building project was to be implemented in accordance to the governmental security instructions, while following the service provider's own management framework. Project itself was managed with Scrum. The project's steering group required the use of Scrum, and this project may be viewed as a showcase of Scrum's suitability to multi-teamed, multi-site, security standard-compliant work.

We also discuss the difficulties of fulfilling strict security regulations regarding both the development process and the end product in this project, and the difficulties utilizing Scrum to manage a multi-site project organization. Evaluation of the effects of the security work to project cost and efficiency is also presented. Finally, suggestions to enhance the Scrum method for security-related projects are made.

***Keywords*-security; Scrum; VAHTI; infrastructure**

I. 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 applied, the drive for security still guides the selection of design patterns, and technological components, as well as the design and development work done in organizations. Increasing diversity of development methods, technology and the environments where information systems are used have prompted the emergence of various security standard. In 2001, the government of Finland begun to issue their own set of security regulation documentation, VAHTI instructions. The regulation is named after the abbreviation of the issuer, the Government Information Security Management Board (Valtiohallinnon Tieto- ja Kyberturvallisuuden Johtoryhmä). Compliance with these security regulations is mandatory for all government agencies since 2014, and they are also enforced to any information system and data which is connected to a government system.

At the same time when the importance of security regulations has increased, the use of lightweight software development processes and methods — commonly called as Agile software development — has simultaneously became a *de facto* standard in the industry. A series of work (see e.g. [1], [2], [3], [4], [5], [6], [7]) has been devoted to develop the security enchanted models for these lightweight methods. However, to the best of authors' knowledge, no empirical case studies in a real-life industrial setting have been previously reported. The study reported in this paper is exploratory, and the research objective is to report the experiences of agile development in security regulated environment and to identify best practices as well as hindrances from the case.

This paper describes a case of a Scrum project with an objective of building an Identity Management system for VAHTI compliant information systems. The target server platform itself can be used to host the agencies' operational systems, and may also be used for hosting development projects (with certain dispensations). The project was executed during the years 2014 and 2015, spanned for 12 months. Project involved an average of 9 persons split into two to three geographically dispersed teams. The amount of teams involved was dependent on the tasks of the current sprint 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[1] (NCSA-FI). The server platform to be built was to be suitable for use for all government

[1]https://www.viestintavirasto.fi/en/cybersecurity/ficorasinformationsecurityservices/ncsa-fi.html

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 actually conducted. As the study revealed that not all the objectives of using 'pure' Scrum were not met, we also make suggestions how the efficiency of Scrum and organization could be improved by introducing certain modifications to the Scrum framework. The modifications, let them be called VAHTI-Scrum, 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 the effects of introducing the security tasks into the cost and efficiency of the project.

II. Background and motivation

Agile development models, such as Scrum, have a well established position in the software industry. The use of agile methods has grown strongly since the mid-2000's, whereas almost all security standards regulating software development processes have their origins back in the 1990's, the time preceding the agile methods [9], [10]. While there are well-documented cases of adjusting Scrum for security work [4], [5], and even achieving formal capability maturity level incorporating agile methods [11], yet still the typical approach, such as the one in the case observed, is to simply start using Scrum without methodology adjustments. In this approach, the security-related tasks are treated simply as items in the backlog. They are given story points, and completed among the other items as best seen fit. Certain security tasks, which cannot be timeboxed because of the inherent uncertainties of the work, or the inexperience of the team, may event be performed and completed as 'spikes' parallel to normal Scrum sprints.

The problem with ad hoc approach is lack of integration of security into the project work: while succeeding in achieving "minimum viable security" by completing the formal requirements, it may not be the most cost-effective way to achieve the goals nor provide the best results for the security of the end product. While careful planning by placing the right tasks into right sprints is possible in this approach, it has the potential lack of proper security requirement management and security task pre-planning. This leads to inefficiencies and, consequently, delays and increased cost.

In this case, a large international ICT service provider was charged with the challenge of building a computing environment complying with Finnish government's VAHTI security guidelines. VAHTI is strictly speaking not a security standard as such, but rather a collection of public security 'instructions', or guidelines, defining the security outlines of various aspects of information systems. In addition to general and public rules, there were several client-specific modifications and additional requirements applied to the building process and project execution. Project itself was required to follow the Scrum framework.

The next subsection provides 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 likely to be applicable also to software safety regulations, in e.g. medical field.

Our argument is that by adjusting the Scrum methodology, both the security cost overhead and the security of the end product can be enhanced. This is achieved by strict adherence to the principle of *security assurance*: security processes are incorporated partly into the Scrum process, as opposed to treating them only as items in the backlog; introduction of new security-oriented roles; also, by incorporating the security features into the end product the full benefit of incremental agile methods may be utilized to achieve better efficiency ratio and, arguably, better end products.

A. Security regulations and standards applied to this case

VAHTI is an open and free collection of the Finnish government's security guidelines, published on the Internet since 2001 [2]. 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 practises throughout the the organizations. As of Spring 2016 (the time of writing this article), the collection comprises of 52 documents. The following instructions were found to be relevant for this project:

- VAHTI 2/2009 "Provisions for ICT service interruptions and emergencies" [12]
- VAHTI 2b/2012 "Requirements for ICT Contingency Planning" [13]
- VAHTI 3/2012 "Instructions for Technical Environment Security" [14].

Only the document 2b/2012 is available in English. The other relevant documents are available only in Finnish, and hence their English titles are unofficial translations. The same applies to much of the VAHTI terminology: official English translations may not exist, or they are inconsistent between documents or 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 standards ISO/IEC 9001, ISO/IEC 27001 and ISO/IEC 27002, 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 affects mostly the financial part of the project, mainly affecting the work load of the scrum master by adding reporting responsibilities. The scrum master was reporting

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

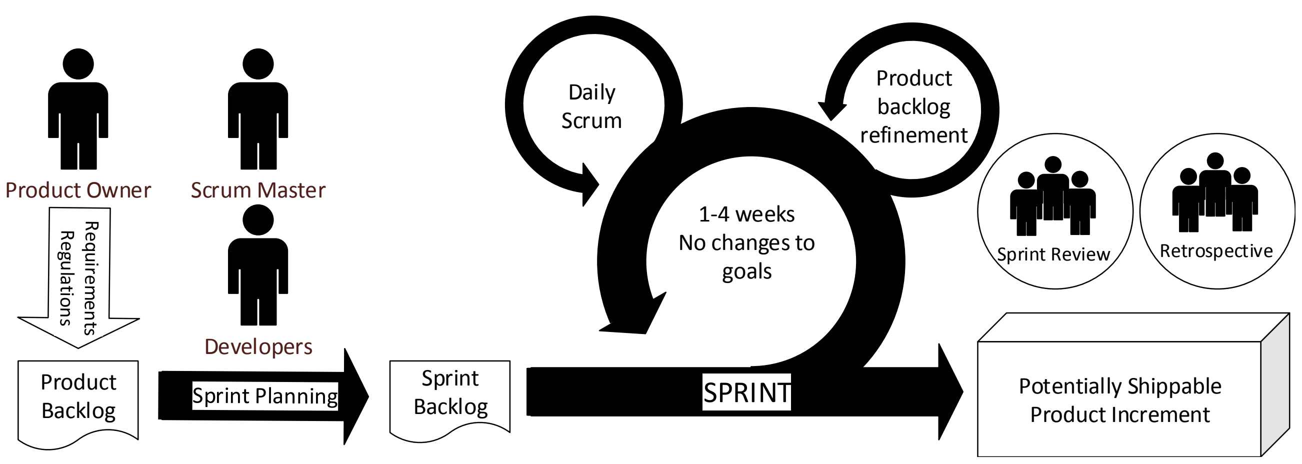

Figure 1. Scrum framework and roles (adapted from [8])

the progress of the project regularly to the steering group, which was part of the corporate PMO, and was also answerable to the corporate financing. The aforementioned company's own proprietary management framework arguably mostly focuses on managing the building company's internal financial risk, but in doing so it converges with general security requirements: it calls for formal risk management, meticulous documentation and well-established and documented processes for development and maintenance, which partially were deemed helpful to the project in this case. The documentation produced by following the management framework processes were in this case directly applicable as evidence providing security assurance for security audits and reviews.

B. Scrum and security

Scrum is a generic framework, typically used to manage software development projects. It suggests that the product to be completed is divided into smaller components, or features, based on 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. Figure 1 presents the basic Scrum sprint structure and key concepts.

The aim of each sprint is to produce a *potentially shippable product increment*, which is presented to the customer, who may decide to deploy this interim version into production. This way, the software product may be utilized to its intended purpose even with a less-than-complete feature set and partial functionality, but with certain key features already present and fully operational.

To accommodate experimental or otherwise more or less unique tasks, such as design, research or prototyping, Scrum has adopted an "enabling mechanism" called a spike. Theoretically this is used e.g. to explore the complexity of a task outside the normal sprint structure, to gain better understanding how it translates into tasks and how its amount of work should be estimated. In practice, the work is often completed during the spike and removed directly from the product backlog, without it ever entering a sprint backlog. Certain security-related tasks, such as penetration testing, fuzz testing, technology research or architectural planning are typical candidates of work items (in Scrum terminology, stories) with uncertain outcome, and therefore likely candidates that call for a spike.

An example of a security-oriented Scrum process is presented in Figure 2.

In this representation the Scrum process is augmented by three major additions:

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) Implementation focused on the *security artifacts*: these items are mostly security-related documents. They consist of security training certificates required from the project team, but most importantly the architecture documentation, risk management plans, test plans, test reports, required log files and other evidence necessary to be presented to the security auditor. The audits produce reports, which are presented to the customer as part of the security assurance.
3) Anticipation and planning of *security-related tasks*. To better illustrate this aspect of security work, they are presented 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

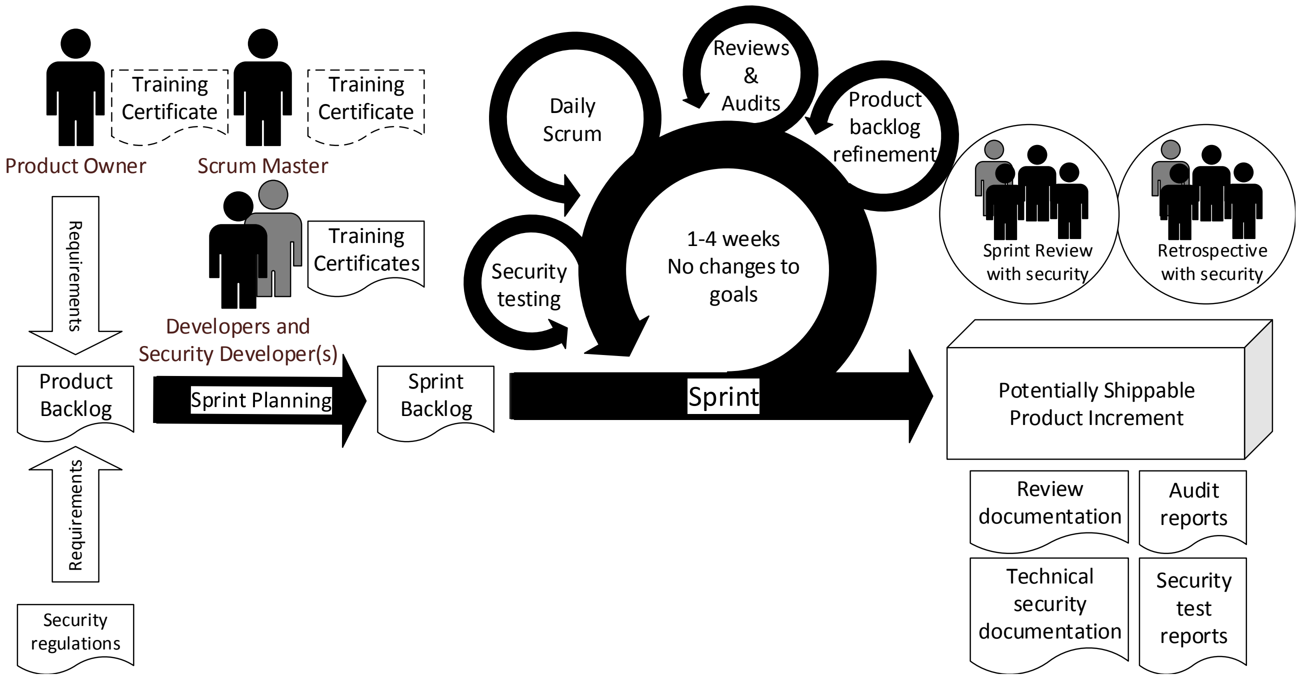

Figure 2. Security-oriented Scrum framework and roles (from [15])

security sprints, the daily scrum may entirely be replaced with the security items.

In the implementations of this model the security testing, reviews and audits are viewed as normal stories in the sprint backlog and executed as part of the daily scrum. Both views are valid, as security tests and audits are in fact part of the product in the case where compliance with security standards and regulations is mandatory. The main shortcoming of difficulty or outright inability to estimate the amount of work involved in these tasks remains still an issue. By emphasizing the importance of security tasks as part of the main product, in comparison 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.

III. Research Process

This study follows a case study design method [16] and a qualitative research approach [17]. For the study, we were looking for a development project that was both using agile methods and fulfilling VAHTI regulations. We decided to focus on 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 done 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.

IV. The Project: building a secure Identity Management Platform

The client, a government agency, wanted to have a VAHTI compliant Identity Management (IDM) platform for their information systems and their administration. The platform itself 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 itself was used primarily to authenticate 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 both security and the client's business. 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 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.

Scrum and other agile methods promote an iterative approach, in contrast to sequential or 'waterfall' approach. Yet, as almost all development work, this project consisted of four main phases, which, in part, were then completed in one or more iterations:

1) *Definition*: synthesis of the requirements, component candidate selection
2) *Design*: architecture design, definition of interfaces, component hardening plans.
3) *Development*: components were researched, modified (i.e., hardened), and installed.
4) *Testing, reviews, audits and acceptance*: security testing, external audits and formal acceptance of the end product as part of the client's information system portfolio.

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.

A. *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), and the developers split into their production teams based on location and occupation.

The organization itself was divided into teams, or ”silos”, just the way many ITIL-oriented organizations are by nature. 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. 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 locations, the meetings were held as a telephone conference almost without exception.

The above 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 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 when their expertise was needed.

B. *Project execution*

The project team was facing a situation where they needed to schedule features with a lot of undefined work and timebox them into three-week sprints. Much of the work related to VAHTI regulations themselves 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 some additional security elements added to the public requirements. This was seen to compensate the requirement of VAHTI instructions for application development [18], which was dropped from the requirement list before the project.

From the beginning, the team's approach to the security tasks was somewhat pragmatic, although rudimentary in terms of Scrum: stories that were found difficult to timebox 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 3, 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. This project applied the Scrum model as presented in Figure 1, with weekly product backlog refinements.

In Figure 3, 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,

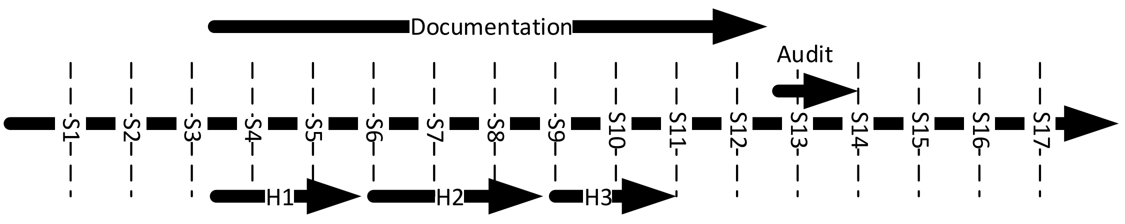

Figure 3. Project structure

although they were executed independently without timeboxing. There were three distinct task types outside the sprint structure:

1) *Hardenings*, performed for each tier or environment of the system under development: Development, Quality Assurance, and finally the Production environment.
2) *Documentation* was an 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 [3]. In this document, there are 224 mandatory requirements listed for the increased security level information systems. Almost all of these require documentation to be verified, although most of the documentation is created in other than the development part of the information system's life cycle.
3) *Audits* were performed based on the documentation and included some physical testing of the implementation.

The project extended over a period of 12 months. 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), this gives a rough estimate of a story point equaling three work days. This sort of conversion may not be meaningful, as story points are primarily used to compare the features (or stories) to each other within a single project. As an overall measure, the story points give an impression of the size of the tasks.

The team was divided into two or three geographically separated locations during the whole length of the project, and due to the team-based separation of the developers, 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 spread from Southern and Western Finland to India. 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.

C. 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 practises were reviewed. All personnel involved with the client was to be security checked by the national Finnish Security Intelligence Service [4]. 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.

VAHTI classifies the information systems into three security levels: basic, increased and high. The server platform itself where the IDM system was installed, was built for the increased security level. Information may be classified into levels of I to IV, where IV is the highest one. Information contained in a system classified for increased security (and audited to provide that level of security) may contain clear-text information up to level III; as this was an IDM system, all the data was decided to be encrypted despite the official classified level.

Project manager estimated that the extra work caused by the regulations was approximately from 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

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

[4]http://www.supo.fi/security_clearances

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.

V. Analysis and discussion

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 3, 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 Section II-B and Figure 2, was taken into use, as the project architect assumed the role of security developer. In practise, most of the security work was done outside the sprints. When the work is done in 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 in the factor is 1.5-2x, caused by the security features, there exists a very large saving potential in rearranging the security work. The bar cannot be lowered, so 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 the similar security feature as spikes even after the first one. This is represented in Figure 3 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 practises 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. Timeboxing 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 to utilize it.

During the interview, the team was quite uniform in their opinions of 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 plannings' effect of keeping the team focused on the tasks at hand, in comparison to e.g. the very long spikes run during the project. In retrospect, the team regretted not utilizing the product owner more already in the beginning of the project, 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.

VI. Conclusions and future work

This paper presented a case of building an infrastructure and setting up an identity management software platform for a governmental customer. The customer had their own set of strict security regulation requirements, the Finnish government's VAHTI instructions. In addition to the governmental security regulations, the ICT service company responsible for building the system was committed to several international ISO/IEC standards, as well as the company's own management frameworks and sometimes complex financial reporting rules. Both the client and the service company's own Project Management Office called for use of Scrum as the project management framework.

The project's personnel was semi-structurally interviewed in a post-project session, and the information was gathered based on their experiences and notes of the project. The parties involved in the project are anonymized, and only publicly available information about the project and the regulations involved was to be disclosed. No financial information was made available, including the team's time keeping which is stored only in the billing system.

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: certain parts of the work were done in spikes outside of the main sprint flow. This was seen in this case to benefit the project, although an iterative and more exploratory approach to those tasks might have proved more benefits in the long term; it is possible that the work done in spikes can be utilized in similar future projects. Despite availability of "Secure Scrum" methods, the approach in this project was more or less an unmodified textbook example of the Scrum method. The only applied modification to the original Scrum were weekly product backlog refinement sessions.

In summary, 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 in the infrastructure projects, although the introduction of growing amount of security and other regulations has increased the demands to 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, however, yet to observe a pure Scrum project where security standards are in a central role. The lessons learned in this project, combined with the suggestions made in this paper, have the potential to provide an excellent starting point for future agile security projects.

Acknowledgment

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

References

[1] K. Beznosov and P. Kruchten, "Towards agile security assurance," in *NSPW '04 Proceedings of the 2004 workshop on New security paradigms*, 2004, pp. 47–54.

[2] X. Ge, R. Paige, F. Polack, and P. Brooke, "Extreme programming security practices," in *Agile Processes in Software Engineering and Extreme Programming*, ser. Lecture Notes in Computer Science, G. Concas, E. Damiani, M. Scotto, and G. Succi, Eds. Springer Berlin Heidelberg, 2007, vol. 4536, pp. 226–230. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73101-6_42

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

[4] D. Baca and B. Carlsson, "Agile development with security engineering activities," in *Proceedings of the 2011 International Conference on Software and Systems Process*, ser. ICSSP '11. New York, NY, USA: ACM, 2011, pp. 149–158.

[5] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, and D. O'Brien, "Scaling agile methods to regulated environments: An industry case study," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13, 2013, pp. 863–872.

[6] P. Pietikäinen and J. e. Röning, *Handbook of The Secure Agile Software Development Life Cycle*. University of Oulu, 2014.

[7] K. Rindell, S. Hyrynsalmi, and V. Leppänen, "A comparison of security assurance support of agile software development methods," in *Proceedings of the 16th International Conference on Computer Systems and Technologies*, ser. CompSysTech '15. New York, NY, USA: ACM, 2015, pp. 61–68.

[8] P. Deemer, G. Benefield, C. Larman, and B. Vodde, *The Scrum Primer: The ligthweight guide to the theory and practice of Scrum*, version 2.0 ed. InfoQ, 2012.

[9] ISO/IEC, *information technology - Security Techniques - Systems Security Engineering - Capability Maturity Model (SSE-CMM) ISO/IEC 21817:2008*, ISO/IEC Std.

[10] ISO/IEC, *Information technology - Security techniques - Code of practice for information security controls ISO/IEC 27002:2013*, ISO/IEC Std., 2013.

[11] J. Diaz, J. Garbajosa, and J. Calvo-Manzano, "Mapping CMMI Level 2 to Scrum Practices: An Experience Report," in *Software Process Improvement*, ser. Communications in Computer and Information Science, R. O'Connor, N. Baddoo, J. Cuadrago Gallego, R. Rejas Muslera, K. Smolander, and R. Messnarz, Eds. Springer Berlin Heidelberg, 2009, vol. 42, pp. 93–104. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04133-4_8

[12] FMoF, “ICT-toiminnan varautuminen häiriö- ja erityistilanteisiin,” 2009, referenced 30th April, 2016. [Online]. Available: https://www.vahtiohje.fi/web/guest/2/2009-ict-toiminnan-varautuminen-hairio-ja-erityistilanteisiin

[13] FMoF, “Requirements for ICT Contingency Planning,” 2012, http://www.vahtiohje.fi, Referenced 17th April 2016. [Online]. Available: http://www.vahtiohje.fi

[14] “Teknisen ympäristön tietoturvataso-ohje,” ref. 18th August 2015. [Online]. Available: https://www.vahtiohje.fi/web/guest/3/2012-teknisen-ympariston-tietoturvataso-ohje

[15] K. Rindell, S. Hyrynsalmi, and V. Leppänen, “Securing Scrum for VAHTI,” in *CEUR Workshop Proceedings, 2015, Vol. 1525, ISSN: 1613-0073*, 2015.

[16] R. K. Yin, *Case Study Research: Design and Methods*, third edition ed. Thousands Oaks, California: SAGE Publications, Inc., 2003.

[17] J. W. Creswell, *Research Design: Qualitative and Quantitative and Mixed Methods Approaches*, second edition ed. Thousand Oaks, California: SAGE Publications, Inc., 2003.

[18] FMoF, “Sovelluskehityksen tietoturvaohje,” 2013, ref. 17th March 2015. [Online]. Available: https://www.vahtiohje.fi/web/guest/vahti-1/2013-sovelluskehityksen-tietoturvaohje