Electronic Research Archive of Blekinge Institute of Technology
<http://www.bth.se/fou/>

This is an author produced version of a conference paper. The paper has been peer-reviewed but may not include the final publisher proof-corrections or pagination of the proceedings.

Citation for the published Conference paper:

Title:

Author:

Conference Name:

Conference Year:

Conference Location:

Access to the published version may require subscription.

Published with permission from:

Identification and Evaluation of Security Activities in Agile Projects

Tigist Ayalew, Tigist Kidane, Bengt Carlsson

Blekinge Institute of Technology, Karlskrona, Sweden

{tigist_eyader@yahoo.com, tabreham.2003@gmail.com, bengt.carlsson@bth.se}

Abstract.

We compare four high-profile waterfall security-engineering processes (CLASP, Microsoft SDL, Cigital Touchpoints and Common Criteria) with the available preconditions within agile processes. Then, using a survey study, agile security activities are identified and evaluated by practitioners from large companies, e.g. software and telecommunication companies. Those activities are compared and a specific security engineering process is suggested for an agile process setting that can provide high benefit with low integration cost.

Keywords: Agile Process, Software security, Development Process, Security Engineering.

1 INTRODUCTION

Today's software development business requires fast software delivery from the development team. For this reason, organizations are transforming their conventional development approaches to agile approaches [1,2]. The key aspect of the agile development is a flexible structure with faster development time that allows handling changes to new requirements easier than the older more rigid processes [3]. Thus transformations are performed in an attempt to increase the effectiveness of software development and are also a result of more online distributed software products and platforms [3]. Agile software development has had a huge impact on how software has evolved worldwide recently [4]. A survey by Forrester a few years ago showed that more than two thirds of the organizations canvassed either already had a mature implementation of agile methods or were midway in implementing such methods [5]. Even though the agile approach is becoming popular, it is reported to have disadvantages related to secure software development [6, 7]. In order to build secure software, security-enhanced processes and practices are needed [23]. Security engineering (SE) processes can be defined as the set of activities performed to develop, maintain and deliver a secure software product; security activities may be either sequential or iterative. Due to constraints such as a lack of a complete overview of a product, higher development pace and lack of documentation inherent to agile processes [6, 8, 9], existing SE processes are difficult to implement in such a setting. Moreover, existing SE processes are designed for a traditional waterfall development approach, i.e. long-

term processes with an emphasis on documentation, which are qualitatively and quantitatively different from agile processes [10].

The growing trend towards the use of agile techniques for building software and the increase in security breaches over the past few years means that it is essential to integrate existing high-profile SE processes with agile processes. Moreover, as there are no SE processes developed specifically for an agile setting, organizations have used existing waterfall SE processes in their agile processes. However, the reliability of the SE processes commonly used in the waterfall model has not yet been evaluated in an agile development setting [3]. Accordingly, existing security activities within waterfall SE processes used in current agile processes are investigated. The identified security activities are evaluated in terms of both cost and benefit. As a result the most compatible and beneficial security activities are identified. Four high-profile waterfall SE processes (CLASP, Microsoft SDL, Cigital Touchpoints and Common Criteria) were investigated in a survey where developers, working in different agile development projects from different locations, have participated.

The rest of the paper is organized as follows. Section 2 presents the background of those SE processes and their activities, which are further used for the survey study. Section 3 presents related work, and Section 4 discusses an empirical study design and validity threats. The background of the respondents and the results from the empirical studies are provided in Section 5, followed by a discussion in Section 6. Finally, Section 7 concludes the paper with the summary of the major findings.

2 BACKGROUND

The agile software development lifecycle is highly collaborative, iterative and includes incremental development. It is originally based on the agile manifesto [24] that advocates breaking tasks into smaller parts with less need of long-term planning.

A specific SE process consists of a set of activities, which are merged and subsequently organized into different phases of a typical software development process. This process typical consists of pre-requirement (PRq), requirement (Rq), design (D), implementation (I), testing (T) and release (R), which will be used throughout our study. These development phases are applied for any software development in general, including agile projects. This section discusses four high-profile waterfall SE processes used for the development of secure software, i.e. the Comprehensive, Lightweight Application Security Process (CLASP), Microsoft's Secure Development Lifecycle (SDL), Common Criteria (CC), and Cigital Touchpoints (CT). The category Others (O) includes further SE processes.

CLASP is a lightweight process for building secure software [13, 15] originating from the Open Web Application Security Project. It originally included a set of 24 independent activities. We exclude 8 security activities from our study, in accordance with the suggestions in [14], i.e. activities not matching an agile development process. As a result 16 security activities are identified that cover most parts of the software development life cycle, except the implementation phase.

Microsoft's experience in securing software products is collected and integrated as the SDL process [2, 15]. It is a heavyweight and tight process for constructing software that helps the software developer resist attacks by addressing security issues repeated-

ly in their products. This process includes a set of 16 sequential activities that cover all parts of the software development life cycle.

CC is an international standard, ISO15408, certified for computer security [9, 17]. It helps developers to specify the security attributes of a product and to evaluate if the products meet their claims [17]. This process includes a set of 7 sequential activities that cover parts of the software development life cycle, except pre-requirement, implementation and testing phases.

CT is a lightweight SE process that helps developers to build security into any software development process [15, 16]. This process includes a set of 9 sequential activities that cover most parts of the software development life cycle, except the pre-requirement phase.

Two other (O) security activities that are commonly used in agile development recommended in [9, 18, 19] but not included in any of the SE processes mentioned above are also identified; countermeasure graphs and pair programming.

Based on these SE processes, initially a total of 50 security activities were found: 16 CLASP, 16 SDL, 9 CT, 7 CC and 2 O. After analyzing the similarities between security activities, by looking at activities having different names but with the same meaning, 9 common security activities were identified as duplicate and, hence, removed from the list. As a result, a total of 41 security activities are obtained which are used in the survey study for further investigation (see Table 1).

3 RELATED WORK

In response to the increased rate of damages caused by security vulnerabilities in software products and the increased use of agile methods in the software industry, some efforts have been made to address software security in agile processes [2]. Previous works in this area focused on theoretical findings and a few studies have used industrial experiences and empirical data.

Researchers have focused on introducing methods for evaluating or integrating security activities in agile processes. Singhal [11] proposes an approach for measuring the degree of agility for different security activities. It determines the degree of compatibility of security activities with an agile method, i.e. a way of measuring the compatibility of security activities with agile software development processes. In addition, a risk removal efficiency factor identifies which security activity can be integrated more efficiently than others. Keramati and Mirian-Hosseiniabadi [2] introduce a method for integrating security activities in an agile methodology to enhance the security of a software product. The method works by defining the agility degree for each security activity and applying an activity integration algorithm with a tunable parameter named the agile reduction tolerance. Siponen et al. [8] describe the requirements for security techniques to integrate seamlessly into an agile process. They demonstrate the requirements selected with an example of an approach for adding security into an agile software development methodology.

A next step is to integrate security methods with an agile model. Ge et al. [12] integrate feature-driven development and high-profile security methods, namely risk analysis, to address the development of secure web applications. Baca and Carlsson identify security activities from three SE processes; Microsoft SDL, Cigital

touchpoints and Common Criteria [3, 9]. Then, they compare those security activities with a specific agile development process, Scrum, which is used in industry. Finally; they produce a security enhanced agile development process by integrating the most compatible and beneficial security activities with agile processes. Beznosov and Kruchten [6] identify and compare security techniques that fit well with an agile development methodology and others that are clear mismatches.

Few studies identify the security issues in an agile process, but Azham et al. identify a number of issues from different security aspects of security in the Scrum methodology [1]. They propose the idea of a security backlog that helps to deal with the security issues in the Scrum methodology. Bartsch presents a report on interviews with practitioners on the effects of agile methods for developing secure software [7]. They also study the implications of security awareness and developer expertise on how security practices are employed.

4 RESEARCH DESIGN AND VALIDITY THREATS

Considering the objectives of the research and the available resources, a survey research method is used to understand practitioners' perception of the current security practices in agile development processes. A survey research method is mainly appropriate for gathering self-reported quantitative and qualitative data from a large number of respondents [20]. A web-based survey questionnaire is used as data collection instrument. A web-based questionnaire is selected in order to obtain information from a relatively large number of practitioners around the world, many of whom we would not be able to contact personally. An inclusion criterion for identifying a target population is used for the survey involving any practitioners in agile software development with security experience, i.e. security related responsibilities or activities. A total of 41 security professionals, who are involved in agile software development, participated in the survey. Two main approaches have been used to get responses for our survey; first responses from people available and willing to participate in the research who meet the inclusion criteria and secondly, by asking the participants to nominate other people who would be willing to participate. The survey questionnaire was designed using an online survey tool (FluidSurvey). It consists of 20 questions, 4 of which concern the respondent information, 6 the current or most recent agile development effort, 7 the identification and evaluation of security activities, and the remaining 3 are open-ended optional questions about agile and security issues. In addition, a coversheet explaining the purpose of the study and procedures, and the description of each security activities were attached to the questionnaire.

The identification and evaluation of the security activities part of the questionnaire was designed using a Likert Scale format. The security activities were evaluated in terms of two criteria: Cost and Benefit. Cost refers to the difficulty in terms of resources, e.g. money, to integrate a security activity as part of an agile process. Benefit refers to the value of continuously using a security activity in agile projects in terms of getting a more secure product. The survey instrument was pre-tested and refined by conducting a formal pilot study with five people who were considered representative of the potential participants of the survey research (practitioners from Ericsson AB with more than three years of experience in agile software development and security).

For the survey we consider the four perspectives of validity and threats as presented in Wohlin et al. [22], i.e. Internal validity, External validity, Construct validity, and Conclusion validity.

Internal validity: designing a readable and understandable survey questionnaire mitigates the threats related to internal validity enhanced by the high completion rate of the respondents (61%). The survey respondents are selected by using various professional networks such as LinkedIn and Academia. This indicates that researchers have no influence on the selection of the subjects for the survey. Furthermore, the sampling error associated with the selection of subject is formally depicted through the use of statistical hypothesis testing.

External validity: The test leaders' control and influence on the respondent answering the questionnaires was limited as the survey was conducted through the web. Also, the samples selected for the study were from diverse agile project setting, e.g. regarding the agile methods, application type, project size, project duration, companies and distribution of the development effort. In addition, the participants have fairly heterogeneous experience in security, agile and waterfall development. Thus, heterogeneous experience of the participants, the diverse distribution of the population and a sufficiently large sample size, show the possibility to generalize the survey findings beyond the selected sample.

Construct validity: The security activities in our research are identified and measured through the survey using close-ended questions. Inadequate preoperational explication of construct threat was avoided by designing the survey instruments in a way to clearly represent the objective of the research. This was done by defining the research aim, the procedures, the evaluation criteria such as Cost and Benefit in the introductory section of the survey. Moreover, the definitions of the security activities are also linked to the survey questions. As a result of this, Mono-operation bias is avoided. The online survey responses were completely anonymous so the evaluation apprehension is mitigated. Another influential risk is the background of the subjects (e.g. experience). However, due to the sufficiently large sample size and the respondents' level of experience in agile development and software security, we consider that the risk associated with the background of the subjects as limited.

Conclusion Validity: A normality test was performed in order to see whether score results are normal distributed or not. As a result, a majority of the score results fulfilled the normality assumption and some of them are approximately normal. Thus, due to these reasons as well as the sufficiently large sample size we consider that the overall influence of the risk related to violation of assumption of statistical tests is limited. The research conclusion is purely based on the independent analysis of the study outcome.

5 RESULTS AND ANALYSIS

5.1 Demographic Data

The respondents were mostly from Europe or North America. Respondents from large, medium and small companies such as Microsoft, IBM, Ericsson, HP and ELC Technologies were participating in the survey. Experience of the respondents in the

agile and waterfall software development had a median of 3 to 5 years and more than 5 years of experience in agile and waterfall development respectively. Furthermore, a large amount of the respondents, 44%, have a development role in current or very recent agile projects. Moreover, the sizes and estimated duration of the agile projects under investigation have a median of 5-10 people and 3-4 weeks respectively. The agile development methods under investigation cover mostly Scrum and Extreme Programming. The respondents were asked whether the current product had been started in a waterfall setting or not. From the result, a majority of the agile projects under investigation, 66%, had not been started in such a waterfall setting.

5.2 Statistical Test

In order to identify the most compatible and beneficial security activities for agile processes, comparisons were made between the security activities. The derived hypothesis for these comparisons focused on finding a significant difference between any two security activities within the same development phase. In order to determine these differences, a one-tail independent sample t-test was used [21]. This is done by checking whether one security activity is preferred against the other in terms of both cost and benefit provided to agile projects. A significant difference is only accepted, if there is at minimum a 95% level of confidence, or at maximum a 5% probability of type 1 error is tolerated, i.e. the incorrect rejection of a true null hypothesis [21].

5.3 Security Activities Evaluation Results

Based on the relative weights of the security activities according to the respondents' rankings, a statistical test was conducted in order to investigate the security activities that are beneficial and compatible with an agile process. From a confidence level point of view, when the respondents judged a security activity as being strongly important (99.9%), highly important (99%) and merely more important (95%) than the others, it was weighted with +++, ++, + respectively as shown in Table 1. In contrast, if the majority of the respondents judged a security activity as being strongly less important (99.9%), highly less important (99%), and merely less important (95%), it was weighted with ---, --, - respectively. The same sign representation is applied for the cost comparison as well, i.e. '+' refers to low cost while '-' denotes high cost. Table 1 presents the identified security activities and their evaluation result in which Group 1 (G1) represents the overall result while Group 2 (G2) represents the results of the respondent group who has 3 or more years of agile development experience.

Table 1. Security activities and their evaluation in agile projects: '+' means preferred and '-' means disliked.

Security Activities	Cost		Benefit	
	G1	G2	G1	G2
Pre-Requirement				
Security Metrics (CLASP)	- -			
Initial Education (CLASP, SDL)	+ +			

	Cost		Benefit	
Requirement	G1	G2	G1	G2
Security Requirements (CLASP, SDL, CT, CC)	++		+++	+++
Abuse Cases (CLASP, CT)		-	---	---
Agree on Definitions (CC)	++	+	-	
Role Matrix (CLASP, SDL)	++	+	--	
Design Requirements (SDL)	--	---	++	++
Identify Trust Boundary (CLASP)	++	+	--	+
Identify Global Security Policy (CLASP)		-	-	
Specify Operational Environment (CLASP)	++	+++	+	++
Identify Attack Surface (CLASP)	-	--	-	-
	Cost		Benefit	
Design	G1	G2	G1	G2
Risk Analyses (CT, CC)	++		+	
Assumption Documentation (CT)	+		--	--
Critical Assets (CC)	-	--	+	+
UMLSec (CC)	+		--	-
Quality Gates (SDL)	++	++	+	
Cost Analysis (SDL)	-	--	+	
Attack Surface Reduction (SDL)	--	---		
Security Architecture (CLASP)	-	--	++	++
Secure Design Principles (CLASP)	++		++	+
Security Activities	Cost		Benefit	
Countermeasure Graphs (O)	++	+++	-	
Requirements Inspection (CC)	++		--	--
Threat Modeling (CLASP, SDL)	--	--	-	

	Cost		Benefit	
Implementation	G1	G2	G1	G2
Static Code Analyses (SDL, CT)	--			
Security Tools (SDL)	+			
Coding Rules (SDL)	++			
Pair Programming (O)	-			
	Cost		Benefit	
Testing	G1	G2	G1	G2
Vulnerability & Penetration Testing (CT)		+	++	++
Red Team Testing (CT)			--	--
Risk Based Testing (CT)				-
Dynamic Analysis (SDL)				-
Fuzzy Testing (SDL)		-	--	--
Code Review (CLASP, SDL)			--	--
Security Testing (CLASP)			++	
	Cost		Benefit	
Release	G1	G2	G1	G2
External Review (CT)	---	--	-	
Repository Improvement (CC)	+		--	-
Incident Response Planning (SDL)	++	+	-	-
Signing the Code (CLASP)	+++	++	-	
Operational Planning and Readiness (CLASP)	--		++	+
Final Security Review (SDL)		-	-	

The following major differences and similarities are observed based on the results in Table 1. In the pre-requirement phase for cost, Initial Education is preferred and Security Metrics is disliked in G1 while in G2 no security activity is preferred to others. In terms of benefit, in both groups no security activity is preferred to the others. This may arise from the fact that most agile projects do not include a pre-requirement phase. Due to this, respondents may not have enough experience to effectively evaluate these security activities.

In the requirement phase for cost in both groups Role Matrix, Agree on Definitions, Identify Trust Boundary and Specify Operational Planning are preferred whereas Design Requirements and Identify Attack Surface are disliked. In terms of benefit, Security Requirements, Design Requirements, and Specify Operational Environment are preferred in both groups while Abuse Case and Identify Attack Surface are disliked.

In the design phase for cost, Quality Gates and Countermeasure Graphs are preferred in both groups. On the contrary, Critical Assets, Cost Analysis, Attack Surface Reduction, Security Architecture, and Threat Modeling are disliked in both groups. For benefit, Critical Assets, Security Architecture and Secure Design Principles are pre-

ferred in both groups while Assumption Documentations, UMLSec, and Requirements Inspection are disliked.

In the implementation phase for cost, no significant differences are observed in G2. However, in G1 while Security Tools and Coding Rules are preferred, Static Code Analysis and Pair Programming are the disliked activities. For benefit, in both groups no significant differences exist.

In the testing phase in terms of cost, Vulnerability and Penetration Testing are preferred against Fuzzy Testing in G1. In terms of benefit, Vulnerability and Penetration Testing is preferred in both groups, Red Team Testing, Fuzzy Testing and Code Review are disliked in both groups. In addition, Security Testing is preferred in G1.

In the release phase for cost, Incidence Response Planning and Signing the Code are preferred in both groups. In contrast, External Review is disliked in both groups. For benefit, Operational Planning and Readiness is preferred in both groups while Repository Improvement and Incident Response Planning are disliked.

5.4 Waterfall SE Processes Evaluation

In this section the activity coverage of SE processes in agile projects for each development phase is presented. The coverage result is derived from the overall result, G1, given in Table 1. As shown in Table 2, CLASP and SDL cover more than two third of the process activities together. This indicates that CLASP and SDL are the most useful SE processes in current agile development. Furthermore, for each development phase, the SE process that covers most is identified. As a result, CLASP is strongly chosen in both the pre-requirement and requirement phases. In the design phase SDL and CC are weakly selected. SDL is strongly chosen in the implementation phase. In the testing phase CT is slightly preferred. Finally, in the release phase CLASP and SDL are weakly selected. Moreover, we also must emphasize that not all SE processes cover the whole agile development lifecycle, only SDL covers all phases.

Table 2: SE Processes activity coverage in agile projects, OC means overall coverage.

SE Process	PRq %	Rq %	D %	I %	T %	R %	OC %
CLASP	75	58	21	0	30	33	34
SDL	25	19	29	63	30	33	31
CT	0	8	13	13	40	17	15
CC	0	14	29	0	0	17	15
O	0	0	8	25	0	0	5

Table 3: SE process evaluation based on the preferred (+)/disliked (-)/in-between number of security activities in agile projects.

SE Process	Cost			Benefit			Total
	+	-	In between	+	-	In between	
CLASP	7	5	4	6	8	2	16
SDL	7	5	4	4	6	6	16
CT	3	2	4	3	4	2	9
CC	6	1	0	3	4	0	7
O	1	1	0	0	1	1	2
Duplicate	- 6	- 2	- 1	- 4	- 4	- 1	- 9
Total	18	12	11	12	19	10	41

From the above identified security activities a further evaluation was performed to identify the number of compatible and beneficial security activities for an agile process. Out of the 41 waterfall security activities used in current agile processes, the following conclusions are drawn. From a cost point of view, 18 security activities are most compatible with an agile process, i.e. they are easy to introduce in an agile environment, 12 security activities are not compatible with an agile process, and the remaining 11 security activities are in-between. From a benefit point of view, only 12 security activities are beneficial to agile processes, i.e. giving high payback to an agile process, 19 security activities are not beneficial and the remaining 10 are in-between. When considering the cost and benefit of the SE processes, CC has most activities preferred by an agile process (86% and 43% respectively) and CT has the least number of activities for cost 33% and in SDL only 25% activities are preferred in terms of benefit. In terms of number of activities CLASP and SDL are positively dominating due to more initial activities present.

5.5 Agile Compatible Security Activities

The most compatible and beneficial security activities for an agile process are presented in Table 5. The goal is to create, from existing SE processes, a new SE process for an agile process, providing high benefit with low integration cost. The selection is performed from the results presented in Table 1. In the selection process, the overall group (G1) and the experienced group (G2) results are considered. The two main reasons for including G2 results are: firstly, the answers of G2 are more reliable and trusted than those of the less experienced group, as those may not have enough experience to evaluate security activities effectively and secondly, from the total respondents, a majority of the respondents (60%) are in G2. Three main selection criteria (C1, C2, and C3) are used in the selection of the security activities as shown in Table 4.

Table 4. Agile compatible security activities selection criteria.

#	Selection Criteria	
	First Consideration	Second Consideration
C1	Preferred for Benefit in G1 or G2	Preferred for Cost in G1 or G2
C2	Preferred for Benefit in G1 or G2	In-between for Cost in G1 or G2
C3	In-between for Benefit in G1 or G2	Preferred for Cost in G1 or G2

By considering the above selection criteria, 16 security activities are identified as both compatible and beneficial to an agile process, as depicted in Table 5. These security activities cover all the phases of a typical agile development project.

Table 5. Agile compatible and beneficial security activities.

Pre-Requirement (PRq)	Requirement (Rq)
Initial Education (CLASP, SDL)	Security Requirements (CLASP, SDL, CT, CC)
Design (D)	Agree on Definitions (CC)
Risk Analyses (CT, CC)	Role Matrix (CLASP, SDL)
Quality Gates (SDL)	Identify Trust Boundary (CLASP)
Secure Design Principles (CLASP)	Specify Operational Environment (CLASP)
Counter Measure Graphs (O)	Implementation (I)
Testing (T)	Security Tools (SDL)
Vulnerability & Penetration Testing (CT)	Coding Rules (SDL)
Security Testing (CLASP)	Release (R)
	Signing the Code (CLASP)
	Operational Planning and Readiness (CLASP)

Below the definitions of the 16 security activities are presented:

- Initial Education (PRq): Everyone on a development project should be aware of the importance of security and the basics of SE which includes; teaching the security concepts, types of security breaches, possible solutions and so on.
- Security Requirements (Rq): Assign security experts, identify and enumerating security and privacy functionality for a given software process.
- Agree on definitions (Rq): The first task for an organization is to define the stakeholders and to agree upon a common set of security definitions, i.e. the definition of the security policies for a software company with the clients as part of the stakeholders' security vision of the IS.
- Role Matrix (Rq): Identifying all possible user roles and their access level to the software.

- Identify Trust Boundary (Rq): Describe the architecture of the system from the perspective of the network, identify data resources that may be used by a program and denote where trustworthy and untrustworthy entities interact.
- Specify Operational Environment (Rq): Document assumptions and requirements about the operating environment, so that the impact on security can be assessed.
- Risk Analyses (D): Security analysts find and prioritize architectural flaws so that appropriate mitigations can begin.
- Quality Gates (D): Create appropriate security and privacy quality measures for the entire software development project, including activities that need to be done for a fulfillment of the requirement.
- Security Design Principles (D): Make the application design harder by applying security design principles and identify security risks in third-party components.
- Countermeasure Graphs (D): A risk analyses method that focuses on identifying security features and prioritizing them.
- Security Tools (I): Define and publish a list of approved security tools to assist the project, i.e. commercially available, open source and in-house developed, and associated security checks.
- Coding Rules (I): Determine the list of unsafe functions and replace those unsafe functions with safer alternatives.
- Vulnerability & Penetration Testing (T): Provides a good understanding of fielded software in its real environment. This is done by simulating real world working conditions and attack patterns.
- Security Testing (T): Find security problems not found by implementation review and catching failures in design, specification and implementation.
- Signing the Code (R): Provide the stakeholder with a way to validate the origin and the integrity of the software.
- Operational Planning and Readiness (R): This includes the writing of user manuals, documenting the security architecture and so on.

6 DISCUSSION

In summary, mainly four high profile SE processes, CLASP, SDL, CC, and CT are used for identifying a total of 50 security activities. Then, 9 duplicate security activities are discarded. As a result, a total of 41 security activities are identified, which are used as input for the survey study. From the survey result, it is suggested that out of the 41 security activities, only 16 security activities are both compatible and beneficial to an agile process. Accordingly, this set of security activities can be considered as a candidate SE process for agile processes.

In terms of cost, a higher number of CLASP and SDL activities is preferred than others. Furthermore, in terms of benefit a higher number of CLASP activities is preferred than others. As a result, in both cases CLASP activities are preferred. These may arise from the fact that CLASP activities are independent while others request a sequential approach to secure software development [15]. Thus, the flexibility of the CLASP activities makes it easier to implement them in an agile process. Furthermore, a majority of the agile projects under investigation were not started in a waterfall

process, and especially, from the agile project that had been started in waterfall, no one uses CLASP as a SE process. This indicates that this process is difficult to implement in a waterfall development setting. However, from the study result it is proposed that CLASP is preferable to use in an agile process model contrary to CC that was chosen as a SE process in a waterfall setting. CC is a lightweight security engineering process that integrates core activities in an existing development process without having any assigned security activities for the pre-requirement, implementation and testing phases of an agile process. This is a reason for CC activities not to be as preferable as CLASP (and to some extent SDL) in an agile environment.

When comparing our work with a previous study by Baca and Carlsson [3, 9] conducted in a real industry setting, we note that both studies investigate SDL, CT and CC. In the previous study a total of 10 security activities compared to our 16 security activities were identified as both compatible and beneficial to agile processes. Since CLASP is not included in the previous study, activities that are selected from the CLASP SE process were not included in the comparison. The comparisons of both studies are discussed as follow.

In the pre-requirement phase, we cannot compare our result with the former study, where this phase is not included. In the requirement phase, in both studies Security requirement and Role matrix are selected. However, in our study more activities are selected as compatible to agile projects, these include: Agree on Definitions, Identify Trust Boundary and Specify Operational Environment. In the design phase, in both studies Countermeasure Graphs is selected. In addition, in our study Risk Analyses, Quality Gates and Secure Design Principles are selected, while in the former study Assumption Documentation, Abuse Cases, and Requirements Inspection are selected. In the Implementation phase, in both studies Coding Rules is selected. In addition, in our study Security Tools is selected where Static Code Analyses is selected in the former study. In the testing phase, Security Testing and Vulnerability & Penetration Testing are selected in our study, while in the previous study only Dynamic Analyses is selected. In the release phase, in our study Signing the Code and Operational Planning & Readiness are selected contrary to selecting Repository Improvement in the previous study. The number of common and different security activities selected as compatible and beneficial to agile processes in both studies is summarized in Table 6.

Table 6. Number of security activities selected as both compatible and beneficial to agile processes in both studies.

SE Processes	Number of security activities	
	Our Result	Previous Study
CLASP	6	0
SDL, CT, CC, O	10	10
Total	16	10
Security Activities selected in both	-4	-4
Differences	12	6

The possible causes for these differences include as a first reason the use of one more SE process, namely CLASP that is not included in the other study, as six of the security activities are identified from this SE process in our study result. A second reason is that because of the major differences that exist between our study and the former study, i.e., the former study base was in one specific telephone company Ericsson AB, while in our study participants from different location and companies participated (small to large companies). As an example Static Code Analyses is regarded as too costly in the present study while being preferred (and also being a tool in use) in the former study, i.e. practical experiences may affect the conducted answers. Also, in the former study interviews were used for evaluating security activities including a small number of participating agile professionals. In our study, an online survey, with a larger number of respondents from different areas, is used to answer the survey questions. Finally the former study addresses specifically Scrum agile method, while our study addresses all the major agile methods, such as, Scrum, XP, FDD, etc.

7 CONCLUSION AND FUTURE WORK

Developing secure software in an agile process needs a SE-process that can address security issues in every phase of the agile development lifecycle; however any of the investigated SE processes was not fully compatible and beneficial to agile projects. Our suggestion in this area is that it is necessary to develop specific agile processes which are different from existing waterfall SE processes. Redundant or too “heavy” activities should be avoided together with insufficient beneficial activities, i.e. not enough benefit compared to the effort invested.

The study conducted is considered as a first step towards the identification and evaluation of security activities that are used in current agile processes. Thus, this paper contributes to the increasing empirical work within the area, and is supposed to provide empirical evidences for practitioners and researchers in the area.

In addition, since the selected security activities are originally developed for waterfall development approach, some of the security activities might need modification in order to adapt with an agile process. We are not investigating new or pure agile SE-processes (but a selection of existing/modified security activities as a base for the agile development). Therefore, the directions for future work primarily include evaluating these security activities that are selected as compatible and beneficial to an agile model in a real agile industry setting. These steps will add value to the findings and gain acceptance in the real agile industry.

REFERENCES

- [1] Z. Azham, I. Ghani, and N. Ithnin, ‘Security backlog in Scrum security practices’, in *5th Malaysian Conference in Software Engineering (MySEC)*, 2011, pp. 414–417.
- [2] H. Keramati and S. H. Mirian-Hosseiniabadi, ‘Integrating software development security activities with agile methodologies’, in *AICCSA 2008. IEEE/ACS International Conference on Computer Systems and Applications*, 2008, pp. 749–754.
- [3] D. Baca, ‘Developing secure software in an agile process’, *Computer Science Department, Blekinge Institute of Technology Sweden*, pp. 129-149, 2012.

- [4] T. Dybâ and T. Dingsoyr, 'What do we know about agile software development?', *IEEE Software*, vol. 26, no. 5, pp. 6–9, 2009.
- [5] D. Bhardwaj, 'Scrumming it up, a Survey on Current Software Industry Practices'.
- [6] K. Beznosov and P. Kruchten, 'Towards agile security assurance', in *Proceedings of the 2004 workshop on New Security Paradigms*, 2004, pp. 47–54.
- [7] S. Bartsch, 'Practitioners' Perspectives on Security in Agile Development', in *2011 Sixth International Conference on Availability, Reliability and Security (ARES)*, 2011, pp. 479–484.
- [8] M. Siponen, R. Baskerville, and T. Kuivalainen, 'Integrating security into agile development methods', in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS'05*, 2005, p.185a.
- [9] D. Baca and B. Carlsson, 'Agile development with security engineering activities', in *Proceeding of the 2nd workshop on Software engineering for sensor network applications*, 2011, pp. 149–158.
- [10] H. Chivers, R. Paige, and X. Ge, 'Agile security using an incremental security architecture', *Extreme Programming and Agile Processes in Software Engineering*, pp. 1325–1327, 2005.
- [11] S. and A. Singhal, 'Integration Analysis of Security Activities from the Perspective of Agility', *2012 Agile India, IEEE*, pp. 40–47, 2012.
- [12] X. Ge, R. F. Paige, F. A. C. Polack, H. Chivers, and P. J. Brooke, 'Agile development of secure web applications', in *Proceedings of the 6th international conference on Web engineering*, 2006, pp. 305– 312.
- [13] 'Category: CLASP Activity - OWASP'. [Online]. Available: https://www.owasp.org/index.php/Category:CLASP_Activity. [Accessed: 08-Aug-2013].
- [14] K. Buyens, R. Scandariato, and W. Joosen, 'Process activities supporting security principles', in *COMPSAC 2007. 31st Annual International Computer Software and Applications Conference*, 2007., vol. 2, pp. 281–292.
- [15] B. De Win, R. Scandariato, K. Buyens, J. Grégoire, and W. Joosen, 'On the secure software development process: CLASP, SDL and Touchpoints compared', *Information and software technology*, vol. 51, no. 7, pp. 1152–1171, 2009.
- [16] G. McGraw, *Software security: building security in*. vol. 1. Addison-Wesley Professional, 2006.
- [17] D. Mellado, E. Fernandez-Medina, and M. Piattini, 'A comparison of the Common Criteria with proposals of information systems security requirements', in *ARES 2006. The First International Conference on Availability, Reliability and Security*, 2006, p. 8–pp.
- [18] D. Baca and K. Petersen, 'Prioritizing Countermeasures through the Countermeasure Method for Software Security (CM-Sec)', *Product-Focused Software Process Improvement*, pp. 176–190, 2010.
- [19] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, 'Strengthening the case for pair programming', *IEEE Software*, vol. 17, no. 4, pp. 19–25, 2000.
- [20] Z. Simsek and J. F. Veiga, 'A primer on Internet organizational surveys', *Organizational Research Methods*, vol. 4, no. 3, p. 218, 2001.
- [21] L. M. Rea and R. A. Parker, *Designing and conducting survey research*. Jossey-Bass Publishers, San Francisco, CA, 1997.

- [22] C. Wohlin, Experimentation in software engineering: an introduction, vol. 6. Springer, 2000.
- [23] J. Allen, S. Barnum, R. Ellison, G. McGraw, and N. Mead, Software security engineering: a guide for project managers. Addison-Wesley Professional, 2008.
- [24] 'Manifesto for Agile Software Development'. [Online]. Available: <http://agilemanifesto.org/>. [Accessed: 08-Aug-2013].