UNIVERSITY OF BOLOGNA

BACHELOR OF SCIENCE THESIS

# State of the art techniques for creating secure software within the Agile process: a systematic literature review

*Author:*
Francesco Maria MONETA

*Supervisor:*
Professor Paolo CIANCARINI

*A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science*

*in the*

dipartimento di Informatica
Scuola di scienze

October 5, 2018

# Declaration of Authorship

I, Francesco Maria MONETA, declare that this thesis titled, "State of the art techniques for creating secure software within the Agile process: a systematic literature review" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

*"Look again at that dot. That's here. That's home. That's us. On it everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every "superstar," every "supreme leader," every saint and sinner in the history of our species lived there–on a mote of dust suspended in a sunbeam."*

Carl Sagan

UNIVERSITY OF BOLOGNA

# *Abstract*

Computer Science
Scuola di scienze

Bachelor of Science

**State of the art techniques for creating secure software within the Agile process: a systematic literature review**

by Francesco Maria MONETA

Agile processes have become ubiquitous in the software development community, and are used by the majority of companies. At the same time, the need for secure and trustworthy software has been steadily growing. Agile software processes nonetheless have proven difficult to integrate with the preexisting security frameworks developed for the Waterfall processes. This thesis presents the results of a systematic literature review that investigates solutions to this problem. The research questions to which the researcher tried to answer are: "which are the latest solutions to enhance the security of the software developed using the Agile process??" and "Which of the solutions discussed have performed best pilot studies?". This study analyzed 39 papers published between 2011 and 2018. The results were ordered according to which exhibited the highest consensus and coded into four sets. The most salient suggestions were: increase the training of the developers, add dedicated security figures to the development team, hybridize security solution from the waterfall processes and add security artifacts such as the "security backlog" and "evil user stories" to Agile.

# *Acknowledgements*

Many thanks to my supervisor, Professor Paolo Ciancarini, and to my co-supervisor, Professor Daniele Russo, for their timely help and their suggestions in the writing of this thesis.
Many thanks to Dott. Marzia Ramponi, she is the reason I am today able to attend University.
Many thanks to Prof. Matteo Zucchi, for making me discover the beauty of Computer Science.
Many thanks to Prof. Emerit. W. Gill Woodall and to Amy Scott, M.D. for their moral support and for the confidence in me they always displayed.

# Contents

*For Audrey, who was always at my side*

# Chapter 1

# Introduction

Agile software engineering methods have become the norm in recent years and their usage has widely outnumbered the waterfall methods (Project Management Institute 2017). A recent survey showed that currently, in 2018, 91% of companies have adopted Agile, up from 82% in 2015 (Dimensional research 2018). Software companies turn to Agile for the shorter time to market, higher productivity, increased quality of the product and heightened engagement and satisfaction from the employees (Cohn 2009). Nonetheless, there is a cost to pay for these improvements. Reduced time-to-market means reduced time to write and maintain the documentation. Because Agile does not put great emphasis on big upfront design, features are presented to the stakeholders sooner. But certain application, with specific non-functional requirements needs, seem to suffer from the reduced preparatory phase. It is the case, in particular, for application that have stringent security requirements. During the years, many frameworks and solutions have been developed to produce secure software. These solutions, like the Capability Maturity Model developed withing the Department of Defense of the United States (Humphrey 1988), were developed to fit into the software engineering main methods of the time, based on the waterfall process. Big upfront design, comprehensive testing and a general top-down approach are key elements of these frameworks. All of these attributes are in clear contrast with the Agile paradigm (Beck et al. 2001) and with the new methods that spawned at the beginning of the new century. New ideas to produce software with a high degree of trustworthiness within the Agile process were required. This thesis tries to present them, discussing the latest and most advanced techniques that the scientific literature has to offer. To perform this study, the "systematic literature review" method was used. It enabled the researcher to acquire, codify and thoroughly present the studies that were meaningful to the this research. The thesis is organized as follows: the first chapter is dedicated to this brief introduction. Following, the background chapter discusses all the topics that are necessary to the understanding of the thesis itself. Chapter three is dedicated to the method and the analysis of this systematic literature review. It contains the information necessary to repeat this study and presents some statistical analysis of the findings. The subsequent chapter, Results, is dedicated to presenting the findings of this study. Each of the two research questions is introduced in a section, in which the data collected from the study papers is thoroughly discussed. The suggestions are ordered in descending order. Priority is given to the solutions that exhibited the highest consensus among the researches. The fifth chapter, Discussion, gives an overview of the methods presented beforehand, debates the the limitation of this study, presents related work and possible future researches. Finally, the sixth chapter, Conclusions, sums up the findings and present a synopses of this thesis.

# Chapter 2

# Background

## 2.1 Software Engineering

The notion and discipline of "software engineering" originated in the 1960's as a way to maximize the quality of the software and increase its reliability by streamlining its development. At the NATO conference on "Software Engineering" of 1968, a discussion took place on how to turn software development, considered too much of an art, into a scientific discipline, by the means of structuring the programs and their execution, using modules and testing (d'Agapeyeff 1969). The classical software engineering methodology that spurred early on and dominated the field until the late 2000's, came from the defense sector. The waterfall model, proposed in an article by(Royce 1970) and developed within the U.S. Air Force, suggests a top-down approach to the development. In a well defined series of steps, the requirements are elicited, the software is designed and coded. In the end testing and deployment conclude the development process.

## 2.2 Agile process

Agile software development methodologies were born in the early 2000 in opposition to plan-driven methods. The latter, perceived as slow and heavy were an unsatisfactory answer to the needs of an ever changing software landscape. Agile has spurred a plurality of different methods, each sharing a set of principles expressed in the Agile Manifesto (Beck et al. 2001). Agile methods privilege "individual and interactions over processes and tools", "Working software over comprehensive documentation", "Customer collaboration over contract negotiation" and finally "Responding to change over following a plan".

### 2.2.1 Scrum

Scrum (Schwaber and Beedle 2002) is the most widely adopted development method based on Agile, being used in more than half the organizations, according to (Collabnet 2018). This method is described as " A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value" (Schwaber and Sutherland 2017). Scrum describes a specific set of artifacts and events to structure the development process and create a product through small but constant series of increments. Moreover, certain figures composing the scrum team are described. Specifically: the Product Owner, whose duties and prerogatives include managing the "Product Backlog" (ordering the items to achieve maximum value, ensure visibility and transparency of the items), the Scrum Master, responsible for the good execution of the Scrum method. This person acts as a mediator between the team and the shareholders and

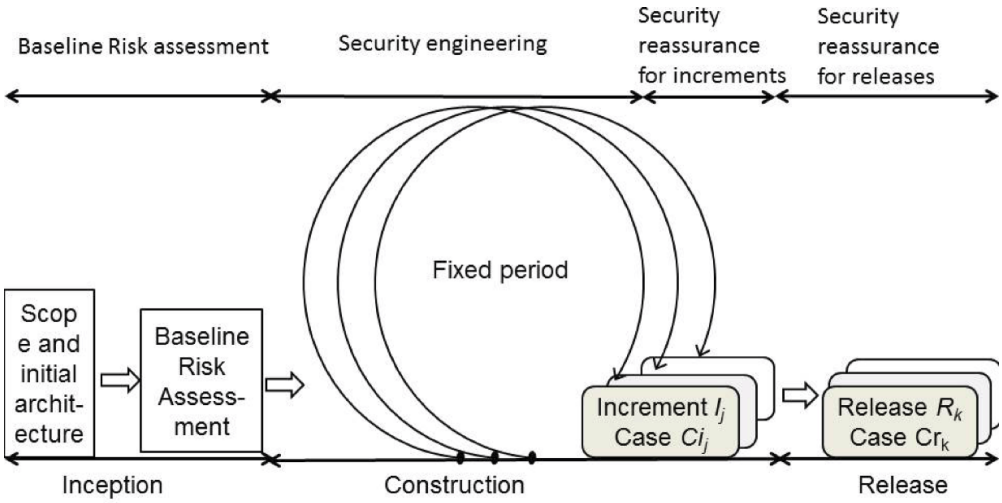


FIGURE 2.1: Iterative nature of the Agile development process, (Othmane et al. 2014)

ensures fruitful interactions between the parties. The Development team should be self organized and cross-functional. A team should be small enough to be deft and agile to minimize the communicational overhead, but still able to produce a value-increment in each iteration. Moreover, it is the team that is held accountable for the development of the artifacts, even for those that have been produced by an single member. An iteration of the process in Scrum is called a “sprint” and it lasts between 2 and 8 weeks. The sprints are designed to provide ad addition of value to the product at every iteration. Because of its iterative nature, Scrum is well suited to adapt to changes in the product requirements.

### 2.2.2 Requirements traceability

Ensuring that all the requirements have been met is not a simple task. Activities of software engineering were developed to trace these requirements, describing their "life" in the context of the life of the project. Thus, we are able to associate at each stage of the development the produced artifacts to the requirements they describe. A novel traceability model is discussed in paragraph 4.1.3 by (Barbosa and Barros Sampaio 2015).

## 2.3 Security

### 2.3.1 What is security

Security is the property of a system or of data that assures its resiliency against improper or malicious use. Security has been historically divided in three main attributes (Brooks 2013): Confidentiality, Availability, Integrity. Specifically, confidentiality assures us that the information we use will remain secret (private) to anybody who does not hold the authorization to view it. Integrity safeguards us on the fact that only those who hold the authorization may change the system or the data. Availability is the property that ensures us that the the information or the system will always be accessible.

Different type of attacks may subvert one or more of these properties. For instance, DDoS attacks will menace the availability of a certain service, without compromising the other two. A man-in-the-middle exploit will instead target the confidentiality. The integrity of a data and system may be compromised if an attacker obtains, through privilege escalation, an administrative account. When developing a secure system, it is important to elicit the requirements that said system will have to satisfy (Bishop 2003). A system exposed to the Internet and that has to handle sensitive information, such as credit card numbers, social security numbers and private records, will have to abide to stricter requirements than a gaming software. Some organization may privilege confidentiality over availability, and vice versa. These considerations will dictate the policy of a system, or the set of rules defining "who is allowed to do what". More technically, a policy defines a set of allowed states in which the system is considered secure. Transitions are also defined, moving the system from a safe state to another one. If the system is allowed to move into an unsafe state, of if a malicious use forces into one, the system is considered non secure.

Enforcing the policy, which is, making sure that the system does not enter unallowed and unsafe states, depends on the mechanisms. The technical mechanisms necessary tend to be complex to implement and, like any software, are prone to include errors.

### 2.3.2 Design principles

As described in the classic work by (Saltzer and Schroeder 1975), there exists 8 security principles that, if used as guidelines, may help to provide a design and an implementation with fewer security flaws. The principles are the following:

- Simplicity or Economy of mechanism: aim at keeping the design as simple and as small as possible.
- Fail-safe defaults: in absence of a specific permission, the standard behavior should be lack of access. Permission should never be granted by exclusion.
- Complete mediation: Every access to any object should be checked for authority.
- Open design: the design should be open and visible. Security “through secrecy”, and thus depending on the ignorance of the potential attacker, is a bad principle and unattainable if the software is widely distributed.
- Separation of privilege: a mechanism that requires two identification methods (keys) will always be inherently safer than another that requires only one.

- Least privilege: each program and user should operate at the lowest privilege level that allows it/him to complete the job.
- Least common mechanism: minimize the amount of mechanism common to more than one user and depended on by all users.
- Psychological acceptability: ease of use should be essential when designing protection mechanisms.

nonetheless, only some of these principles have become staples of security practices and can be found on modern widespread and complex software. As noted in (Smith 2012), three principles do not play a role anymore in nowadays software landscape. Simplicity, complete mediation, and psychological acceptability, due to market requirements, obsoleteness and human behavior, respectively.

### 2.3.3 Touchpoints

While seldom mentioned in the papers scrutinized in the production of this systematic literature review, many if not most of the ideas proposed can be traced to the work of Gary McGraw. In his book (McGraw 2006), McGraw counters the issue of security reaction, proper of organizations that encountered losses due to lack of security in their software, by "building security in". A set of seven "touchpoints" or best principles are suggested as one of the pillars of creating secure software. These appeared initially on (McGraw 2005) and since then have been widely adopted "by the US National Cyber-Security Task Force report, by Cigital, by the U.S. Department of Homeland Security, and by Ernst and Young" (McGraw 2006).

In his work, McGraw proposes these best practices to fill the gaps between the "state of the art" security ideas and the actual industry practices. These touchpoints are said to apply to any software development process, regardless of the methodology. This agnostic way of implementing security is stressed as critical, since organizations will not be prone to subvert the proven methods on which they rely to produce software. Instead, the "touchpoints" are ensured to be easily portable to any agile or waterfall process, and thus can be migrated with minimal disruption. While all important and highly recommended, the author orders the best practices from most to least effective, so that organizations may prioritize their implementation. The touchpoints are in order:

1. Code review
2. Architectural risk analysis
3. Penetration testing
4. Risk-based security tests
5. Abuse cases
6. Security requirements
7. Security operations

These 7 practices provide an holistic approach to security and comprehend both self described "destructive" and "constructive" practices. The first ones are described as activities to "break software", attacks and exploits, designed to highlight flaws and

Touchpoints

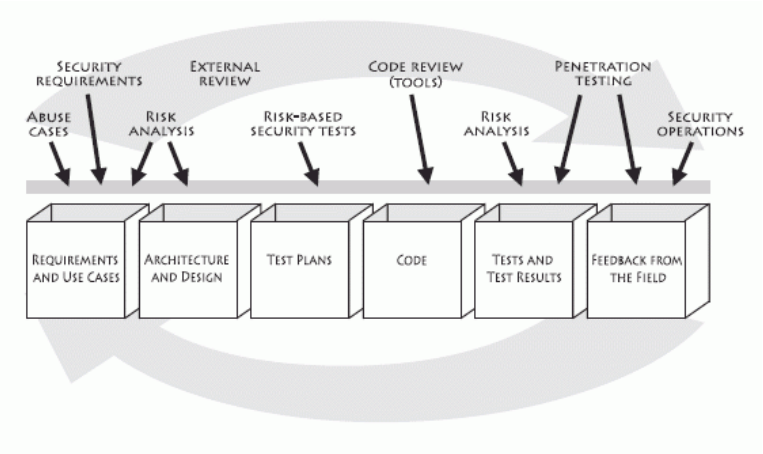


FIGURE 2.2: Touchpoints chart as shown in (McGraw 2006)

vulnerabilities during development. The second ones, defensive in nature, are based on safe design and functionality.

By reviewing the results of the multiple studies taken into consideration in this systematic literature review, it is clear that these touchpoints were, and remain, highly influential in dictating the strategies to produce secure software.

### 2.3.4 OWASP

The Open Web Application Security Project (*The OWASP foundation* n.d.) is a non-profit organization founded in 2001 with the aim of improving the security of software. During the years, many projects, documentation and tools were developed within it. Many suggestions found in results of this systematic literature review are directly derived from OWASP recommendations. Thus, in this background section of the research, the most influential ones are reviewed.

The most famous document, produced in various iterations since 2010, is the "OWASP top 10" (*OWASP top 10* 2017), which expresses the consensus of the IT community on the ten "most critical security risks to web applications". Each of the risks gets a score on multiple concerns such as: Exploitability, Prevalence, Detectability, and a technical value. Attack scenarios and tips on how to prevent the vulnerability are also present, to provide real-world advice and effective guidance to developers. It is seen as a stepping stone for any organization wanting to implement a change in the software culture and to produce secure code.

Another well known and recurring tool that gets mentioned in the research papers is the "OWASP Dependency-Check" (*OWASP dependency check* n.d.). At its core, it is an application that automatically checks the dependencies used by the software. The results are then run against a database containing all the publicly known vulnerabilities for the libraries. According to (Williams 2014), while many companies are increasingly aware of the need for the non functional requirement of security in their custom code, few of them realize the inherent danger represented by insecure libraries. This under-appreciated danger affects most of the software, because modern complex software products strongly rely on them. Up to 80% of the total number of lines of code may be composed by libraries.

### 2.3.5 Microsoft Security (Software) Development Lifecycle

Microsoft Secure Software Development lifecycle is a process developed for waterfall methodologies to "build more secure software and address security compliance requirements while reducing development cost" (*Microsoft SDL* n.d.).

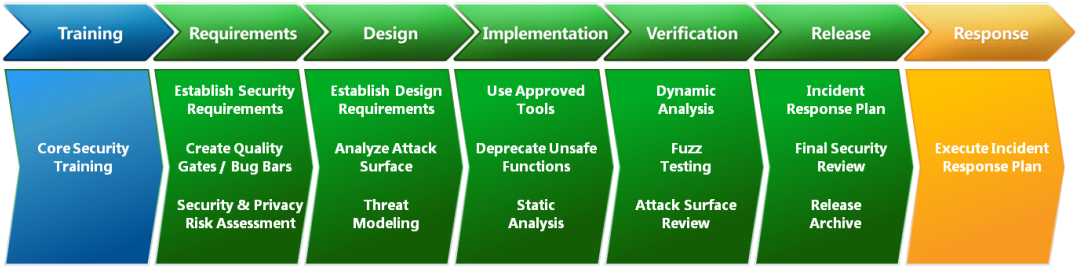


FIGURE 2.3: Microsoft SDL proprietary security process, waterfall pictured.

Microsoft also developed a version dedicated to Agile development (*Microsoft SDL for Agile* n.d.). Facing the problematic that, as this research will show in chapter four, many other researchers had to tackle when hybridizing waterfall and agile processes, Microsoft opted for a novel solution. The company re-organized the different SDL practices by the number of times these gets do be executed. The practices, pictured in figure 2.3, are divided in: "every sprint", "bucket" and "one-time" practices, according to the frequency with which they have to be completed. Moreover, Microsoft stresses the importance of training, even before approaching SDL. Core security training is a prerequisite for both SDL and SDL for Agile. This suggestion will be recurring in the research papers and will be further discussed in chapter 4.

### 2.3.6 Decomposition of Security

Presented for the first time in the article by (Chenxi Wang and Wulf 1997), the "decomposition of security" is a technique that will get further discussed in the findings of this research. It produces an evaluation of the security strength of a system by decomposing it into its core components. The breaking-down phase is formed by four steps:

1. the identification of a security related goal (root)
2. the identification of components vital to the goal (nodes)

3. iterative decompositions of these nodes until no further simplification can take place

4. end the process when the leafs are independent from each other and can be assigned a security score

A security score is assigned to each of these resulting components and then, the aggregated value is computed to form the final estimate. This was a pioneering security metrics development approach, novel in the security field when presented at the time.

# Chapter 3

# Research method

There exists many research methods to conduct a study in the Software Engineering field, as described in (Basili, Selby, and Hutchens 1986). The chosen method to conduct this thesis was the Systematic Literature Review, a research process originating in the medicine field. It provides a broad but accurate view of all the solution to a problem, which can then be analyzed, compared and interpreted. It consists of 3 main stages: planning, conducting and reporting the data. Initially, during the

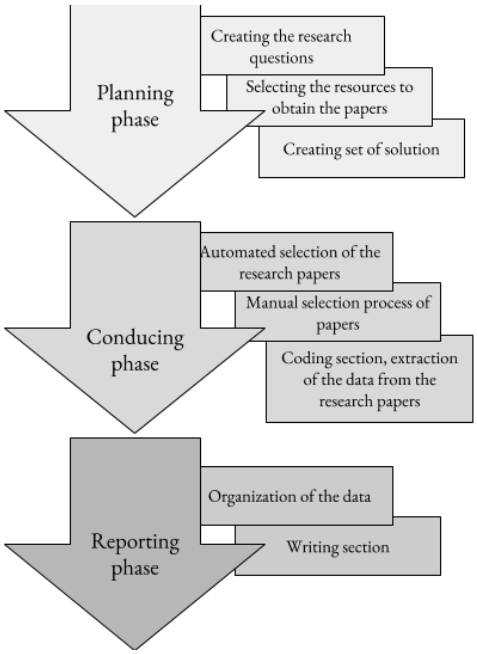


FIGURE 3.1: This charts shows the three main phases of this systemic literature review. Each phase was composed by number of subsections, shown in the rectangles on the side.

planning phase of this study, the research question were developed, making sure that these were narrow and precise enough to make the SLR manageable. Developing a strategy for conducting the research in advance was not only advisable but necessary to avoid bias in the research questions and in the coding phase. Once the scope was defined, a selection process and a strategy for the relevant paper was devised, as well as inclusion and exclusion criteria. To perform the data extraction phase, groups were created to incorporate the solutions. When conducting this SLR, the papers were selected using the aforementioned strategy, then the data was extracted and coded for ease of access and control. Solutions were summarized and categorized, before being actually reported on this review.

## 3.1 Research questions

Two research questions were studied in the completion of this thesis. These were:

RQ1: Which are the latest solutions to enhance the security of the software developed using the Agile process? RQ2: Which of the solutions discussed have performed best pilot studies?

## 3.2 Selection Process

The process of obtaining the review papers was preceded by the creation of a strategy, as highlighted in (Kitchenham 2004). The steps followed were:

1. Automated search querying research databases
2. Selection of relevant papers based on title
3. Full text review and quality assessment

Step 1 started by querying four different databases: IEEE, "Web Of Science", ACM and Scopus. The research strings, developed beforehand to avoid bias, evolved after a series of trials until the results were satisfactory.

Furthermore the research strings had to be adapted to the different standards and dialects of the peculiar front-end systems while maintaining the same semantic expression. The researcher concluded not to incorporate any paper older than 2011 for two reasons: firstly because this thesis wanted to propose and analyze only the most recent solutions developed in the academia. Secondly, it was necessary to maintain a scope that resulted manageable for a single researcher to work on. The inclusion and exclusion criteria were developed before the actual selection process, but were subject to improvements and changes during that phase. The initial search prompted 156 results. These were then inspected to reveal duplicates and false positives. A final batch of 39 relevant papers was then reviewed and extracted. Moreover, a quality value was assigned to each paper. This qualitative screening rated each and every paper "low", "average" or "high". This aided the researcher in compiling the results and establishing if a consensus on certain practices had been established.

## 3.3 Data extraction and coding

The data extraction phase was performed by mapping the findings of the papers on four different answer sets, relevant in answering the two research questions. Thus, each paper was completely analyzed to extrapolate its significance and its data. The "findings" section of the studies was always the part that required the most attention. Each and every suggestion was cataloged with in depth explanations. Subsequently, through a series of summarizations, the information was distilled to a form detailed enough to allow for discrepancies of method between similar studies, but general enough to allow for the coding phase to take place. The subdivision of the papers into four groups could then take place. Each solution presented in the papers was mapped into the corresponding group or groups, defined a priori. Confirming the quality of the initial planning phase, each study mapped directly to at least one set. This reassurance was essential. Had the papers shown ideas that were completely

impossible to reduce to one of the solution sets, it would have meant having to redesign the research from the ground up to take the additional non-mapped suggestion into consideration. Various papers were instead mapped on multiple sets, thus creating a complex web of researches to analyze in order to find the consensus on the best strategies.

TABLE 3.1: List of research papers belonging to each group

| Artifacts of Agile | Human Factor | Agile Process | tools |
|---|---|---|---|
| (Maier, Ma, and Bloem 2017b) (Imran Ghani and Jeong 2014) (Baca, Boldt, et al. 2015) (Bartsch 2011) (Barbosa and Barros Sampaio 2015) (Epstein 2008) (Rindell, Hyrynsalmi, and Leppänen 2016) | (Imran Ghani and Jeong 2014) (Baca, Boldt, et al. 2015) (Poller et al. 2017) (Adelyar and Norta 2017) (Caldwell 2015) (Terpstra, Daneva, and Chong Wang 2017) (Alnatheer et al. 2013) (Oyetoyan, D. Cruzes, and Jaatun 2016) (Camacho, Marczak, and D. Cruzes 2016) (Bartsch 2011) (Tigist, Kidane, and Bengt 2013) (Barbosa and Barros Sampaio 2015) (Epstein 2008) (Felderer and Pekaric 2017) (Loser and Degeling 2014) (Choliz, Vilas, and Moreira 2015) (D. S. Cruzes et al. 2017) | (Maier, Ma, and Bloem 2017b) (Baca, Boldt, et al. 2015) (Balasubramani et al. 2012) (R. Savola, Frühwirth, and Pietikäinen 2012) (Keramati and Mirian-Hosseinabadi 2008) (Tigist, Kidane, and Bengt 2013) (Franqueira et al. 2011) (Baca and Carlsson 2011) (Choliz, Vilas, and Moreira 2015) (Maier, Ma, and Bloem 2017a) (Rindell, Hyrynsalmi, and Leppänen 2015) (Siponen, Baskerville, and Kuivalainen 2012) (A. F. B. Arbain, Ghani, and Kadir 2014) (Othmane et al. 2014) (Renatus, Teichmann, and Eichler 2015) (Sonia, Singhal, and Banati 2014) (Hutchinson, Maddern, and Wells 2011) (Rindell, Hyrynsalmi, and Leppänen 2016) | (Mackey 2018) (Raschke et al. 2014) (Munetoh and Yoshioka 2013) (Terpstra, Daneva, and Chong Wang 2017) (Felderer and Pekaric 2017) (Renatus, Teichmann, and Eichler 2015) (Sonia, Singhal, and Banati 2014) (D. S. Cruzes et al. 2017) (Hutchinson, Maddern, and Wells 2011) (Kumar et al. 2012) (Bansal and Jolly 2014) |

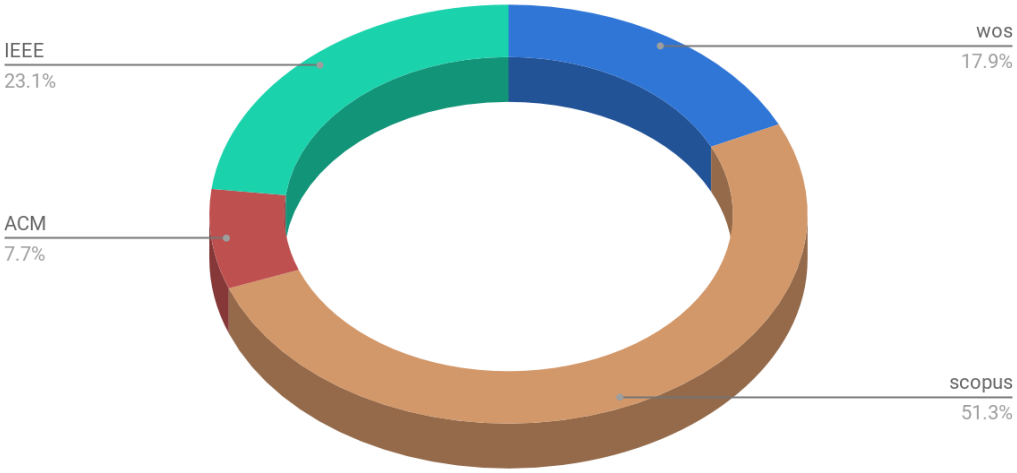


FIGURE 3.2: Databases included in search, and number of matched articles. The research papers were obtained by querying 4 different databases. Here the graph shows the percentage of papers included in the literature review that came from each of these.

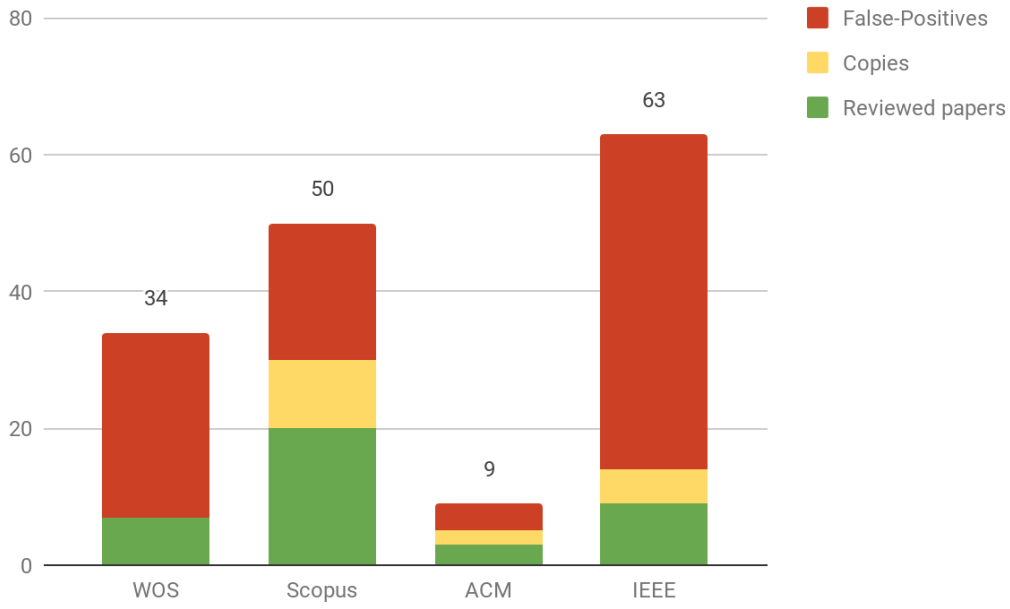


FIGURE 3.3: Not all the papers obtained from the databases were useful. A number of papers were removed during the manual phase of the selection process. This chart illustrates how many studies from each database were either false-positive and thus removed (red), copies (yellow) and papers that made it into the SLR (green). Note: "Web Of Science" bar shows no copies because the output of this database was the first one to be analyzed.

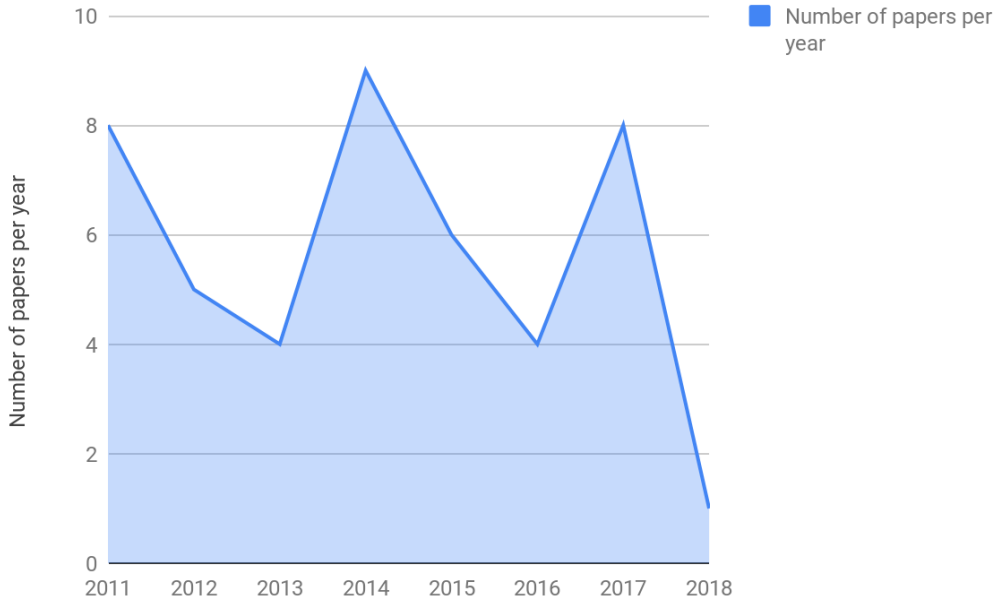


FIGURE 3.4: This line chart plots the number of papers included in the research that were published in each year.

## 3.4 statistical analysis

4 distinct databases were queried: IEEE, ACM, Scopus and Web Of Science, prompting 63, 9, 50 and 34 results respectively. Figure 3.2 charts the years in which the papers taken into consideration for this study were published and displays their number. Figure 3.3 illustrates how many papers ended up being used, how many were copies and how many were included in the automatic query results but were non-pertinent to this systematic literature review.

# Chapter 4

# Results

## 4.1 RQ1: Which are the latest solutions to enhance the security of the software developed using the Agile process?

To answer this question the data extracted from the papers was mapped on 4 different groups/sets, which were able to fully express and synthesize the studies' results. The sets considered were the following:

1. Solutions addressing artifacts of the Agile method
2. Solutions addressing the human factor (Training, professional figures)
3. Solutions addressing the agile process itself (frameworks, organization, activities)
4. Solutions addressing the tools

Papers that offered multiple solutions were mapped in each of the relevant groups. Although some of the research paper provided solution that mapped on multiple sets, most of the them took the "agile-security" problem and answered it by narrowly addressing certain issues, while neglecting the other aspects of the development. This lack of a overall view was furthermore exacerbated by the endemic absence of real world data, a problematic acknowledged by most of the authors. The studies that did involve industry players reported mostly mixed results, perduring the view that a "silver-bullet" technique may not be found. Moreover, the solutions that implemented security principles borrowed from the Waterfall development often failed to scale and adapt to the Agile Development cycle, thus prompting delays and running in over-cost. The findings are loosely organized in descending order, presenting first the suggestions that gathered the greatest consensus, then followed by the remaining ones of their group. Subsequently another group is presented. Proposal that failed to appear in more than one paper and that were given a "low" score in quality are not presented individually.

### 4.1.1 Human factor

Although being second in number of citations in the papers group-wise, the "human factor" set was the one that exhibited the highest consensus. Virtually all the suggestion in the category depicted a grim panorama in the agile development community. The lack of training and understanding of security issues, at each and every level of the development team, was viewed as the root cause of the security vulnerabilities in the software being developed. The suggestion raised was to invest profoundly on seminars and training for the team, thus increasing the Security Awareness of the members. How this training should be performed and who should received was

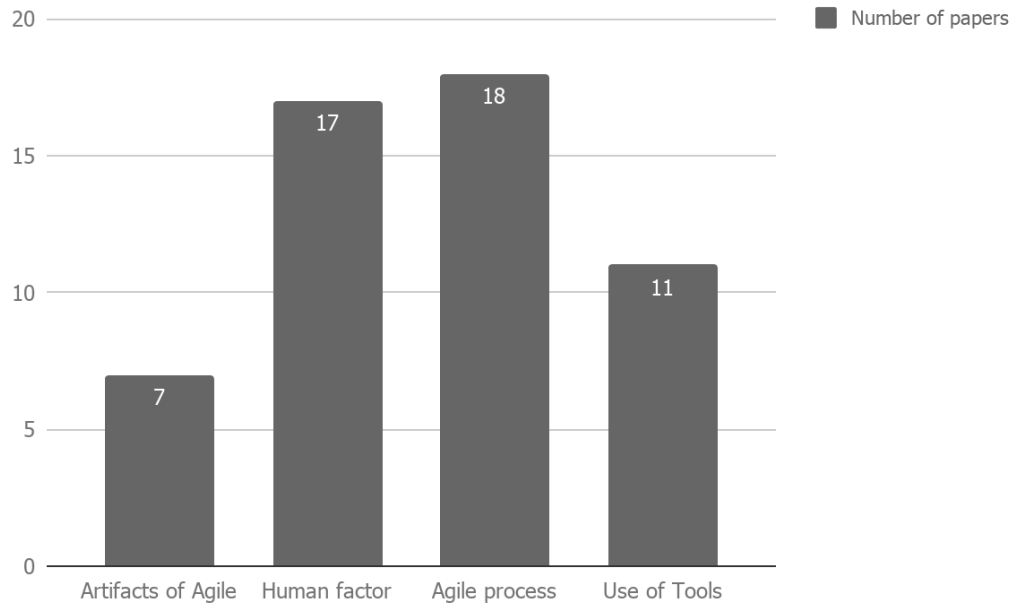


FIGURE 4.1: Sets mapping on the studies. It is easy to see that

matter of debate.
Specifically, (Poller et al. 2017) studied the improvement of security through workshops and security consultants in a multinational company using Scrum and Kanban as methods for the production cycle. The company's teams were self organizing and the management could steer the focus towards security only indirectly, through quality indicators. moreover, neither the company nor its products had ever experienced losses related to malevolent attackers. Thus, security was perceived of little value, and directly stacked against other non-functional requirements (such as performance, usability and scalability) and against the features which hold direct value for the company. Although the security training was perceived as beneficial and raised the awareness of the teams, few changes were effectively incorporated into the development life-cycle. Individual developers were in charge of making sure that security was integrated in the features they were working on. But often, these activities were executed only superficially, due to the time constraints and pressure to roll-out the product. Finally, the task of solving issues was usually assigned to the most experienced developer in that area, limiting knowledge exchange between team members. This issue was highlighted by (Oyetoyan, D. Cruzes, and Jaatun 2016), which argued that an effort should be made to create an effective environment that makes replication of software security successes possible among teams. In the study, it was also noted that the area in which coaching efforts should be focused the most was secure design training. This came as no surprise, as the lack of attention to design and architectural issues is a serious limitation of the agile approach (Rosenberg and Stephens 2003).

(Adelyar and Norta 2017) consisted in a case study of the security activities of agile based on (Saltzer and Schroeder 1975) 8 principles. The interviews with developers pointed to little security awareness as the main cause of flaws in agile development, similarly to what indicated by (Caldwell 2015) and (Terpstra, Daneva, and Chong Wang 2017). in the studies by (Camacho, Marczak, and D. Cruzes 2016)

and by (Bartsch 2011), a series of interviews with IT professionals in different multinational companies were performed, and highlighted similar issues, although no suggestion were made. It was also noted that while experienced developers tend to consider non-functional requirements, like security and performance, more than junior professional, they failed to spread their attention and care to the other members of the team. This reinforces the idea presented by (Oyetoyan, D. Cruzes, and Jaatun 2016) that without an environment that aids the replication of security knowledge between teams and members of the same group, the development of software with high non-functional requirements standards is left to the non-homogeneous expertise of the development teams. The management and the stakeholders were also considered, bringing up the issue related to the perception of security not as a desirable non-functional requirement , but only as a nuisance and a cost to be curtailed. The attitude towards security from these parties had to shift, according to the paper if security was to be improved.

(Loser and Degeling 2014) recommended the introduction of “hygiene requirements”(salary, workspace qualities and conditions or work-life balance) to improve the developers attention to security. It was perceived that training alone resulted insufficient to ensure a high degree of security in the development process. Thus the introduction of a hygiene requirements to ensure that non-functional requirements would get the amount of attention needed. This article approached the problem of security from a completely different point of view, focusing on the psychology of the developers and their level of satisfaction.

A different approach to the problematic of developers expertise highlighted in the studies suggested the addition of one or more professional figures to the development team. While the addition of experts was perceived to substantially increase the overall security of the product, questions about the costs of such a solution were discussed. Moreover, agile philosophy places much emphasis on the concept of “cross functional team”. The addition of a security dedicated professional with a specific role would seem to violate.

Taking inspiration from the concept of the "security principles" from the classic study by (Saltzer and Schroeder 1975) and from Microsoft SDL, (Imran Ghani and Jeong 2014) proposed the addition of a "Security Master" figure, to be added to the Scrum process. This new role, in charge of the newly created "security backlog" (see paragraph 4.1.3) would increase the degree of agility of the process. (Epstein 2008) also proposed the introduction of a "security master". However a limitation noted by the study to this solution was the "shortage of professional security experts to perform this role". The addition of a professional figure was also suggested by (Alnatheer et al. 2013), such as a software engineer that should consider security at every step of development. In this last study, the addition of a security professional to the team gained more approval amid IT professionals than the idea of "developing with security in mind", although adding a security engineer to the process was considered more costly.

A more extreme approach was proposed in the research papers (Baca, Boldt, et al. 2015) and (Choliz, Vilas, and Moreira 2015). In (Baca, Boldt, et al. 2015), a group composed by a security manager, a security architect, a security master and a penetration tester was integrated with the agile development team. Their responsibility were discusses in depth. Respectively, the "Security Manager" handled the traditional features connected to security, such as ISO-certification and legal aspects connected with the development. Moreover, he was responsible for the prioritization of

the requirements in the product development. The Security Architect was responsible for transforming the document presenting the general non-functional requirements into a more technical depiction. The Security Master was instead accountable for the security features during development. This figure was also in charge of the risk analysis, an activity meant to estimate the likelihood and the the degree of negative consequences of attacks. Finally, the Penetration Tester, being part of the quality assurance team, conduced exploratory testing and verified that the system could withstand improper malicious use. This approach proved extremely effective in reducing the number of issues present in the product and increase its trustworthiness, but at the cost of an increased spending for dedicated personnel (see paragraph 4.2.2). A similar effort was studied by (Choliz, Vilas, and Moreira 2015). In this case, the security team was not integrated with the other development groups, but was granted a high degree of freedom and independence. It was noted, nonetheless, that this went directly against the Scrum directive of cross functional teams. The Scrum guide states clearly that "Development Teams do not contain sub-teams dedicated to particular domains like testing or business analysis" (Schwaber and Sutherland 2017).

### 4.1.2 Solutions addressing the Agile process itself

Hybridizing waterfall security principles with agile development was a common proposed solution. The software development processes that were ported and hybridized with agile came from other frameworks like Microsoft SDL, CLASP by OWASP and the Capability Maturity Model. In this section of the research the approaches taken by the researches and the resulting processes created will be discussed. Finally, some standalone event addition are presented.
The most cited framework by the studies was Microsoft Secure Software Development Life cycle (*Microsoft SDL for Agile* n.d.). Its success was explained in (Choliz, Vilas, and Moreira 2015) by its effectiveness, usability and because it organized security activities in categories according to how often these are executed. The theoretical research by (Rindell, Hyrynsalmi, and Leppänen 2015) argued that SDL, in conjunction with Scrum or Extreme Programming can easily enhance security and thus "that agile development is readily adaptable to even the most strict security requirements". Other studies looked into improving the solutions proposed by multiple members of the industry. (Baca and Carlsson 2011) evaluated SDL, Cigatel touchpoints and Common Criteria to effectively produce an security enhanced development method, providing the maximum benefit without hindering the degree of agilness of the development. the results showed that these frameworks do not scale well into agile the context. Microsoft SDL (*Microsoft SDL for Agile* n.d.) was said to have "the largest negative effect on an Agile development process", in clear contrast with other studies that instead embraced its model. "Dynamic analyses" was the only favorable activity viewed by the review panel, while "Threat modeling", "specific tools", "Cost analyses" and "Incident response" activities were either considered too costly or incapable of scaling to agile methods, hindering the process. Cigatel Touchpoints activities (McGraw 2006) were praised in the requirement, design and implementation phase, but were considered problematic during the tests phase, achieving minimal improvements at a high cost. Finally, Common Criteria (Keblawi and Sullivan 2006) had troubled hybridizing due to the assumption of an upfront design resulting in an "lengthy implementation phase". The resulting "Agile Security Process" combined the most compatible and beneficial activities, thus

providing the best theoretical cost-effective solution. In a similar study, (Tigist, Kidane, and Bengt 2013) compared the aforementioned development methods, SDL, Touchpoints and Common Criteria, with CLASP (*CLASP principles* n.d.), a set of best practices developed by the OWASP organization. The study then provided a list of the most effective activities to integrate into the agile method, ordered by the development phase in which they should be applied. Initial education was proposed as a "pre-requirement" of the development, hinting to the need for a certain know-how when building secure application. For the initial requirement phase, they suggested to implement "security requirements", similarly to what (Baca and Carlsson 2011) suggested, together with the following activities: "Agree on Definitions", "Role Matrix", "Identify Trust Boundary" and "Specify Operational Environment". During the design phase, the beneficial activities highlighted were "Risk Analyses", "Quality Gates", "Secure Design Principles" and "Counter Measure Graphs". Testing highlighted "Vulnerability and Penetration Testing" from the Touchpoints and "security testing" from CLASP. During the implementation phase, the activities that generated a consensus were the use of "Security Tools" and "coding rules", both from SDL. Finally, for the release phase, "Signing the Code" and "Operational Planning and Readiness" were the suggested activities.

The research by (Baca and Carlsson 2011) produced another version of a security enhanced agile process. The authors analyzed the security activities proposed by Microsoft SDL, OWASP touchpoints and by the Common Criteria recommendations and then interviewed a group of professional to obtain feedback on them. The resulting activities were supposed to be the most beneficial at the least cost. The activities during the requisites phase of the development were: "Security Requirements" and "Role Matrix". Similarly, the only activities suggested during the design phase were: "Static Code Analyses" and "Coding Rules". During design instead, the proposed activities were: "Countermeasure graphs", "Assumption Documentation", "Abuse Cases" and "Requirement Inspection". "Repository Improvement" was considered beneficial after the release of the product. Finally, during the testing phase, "Dynamic analyses" of the program was seen as cost effective. It is interesting to note that the suggested activities for the requisite and design phases appear also in the similar research by (Tigist, Kidane, and Bengt 2013) hinting to a certain consensus among the researchers. No empirical test-case was carried out.

The perception that these waterfall addition would impact the “agilness” of the team was widely discussed. These solution were thus considered a tradeoff between the need to implement security and the will to adhere to the Agile principles. Some research approached this tradeoff in a quantitative way.

(Sonia, Singhal, and Banati 2014) introduced core security activities to the Agile "Extreme Programming" process based on CLAPS. To choose which activities performed best and were most "agile-effective", the researchers created an "agility matrix", where each activity from CLASP was weighted and received a score for its "agilness". Afterwards the researchers produced an "integration matrix", by matching the beneficial CLASP activities to the phases of the Extreme Programming process to which they could be associated. The resulting process could thus be adjusted on the spot by selecting only the activities with a degree of agility above a certain threshold. The researchers also developed a tool, called TISA-XP, that allows to graphically select this threshold degree of agilness, and outputs the resulting selected activities.

Other frameworks with associated tools to quantitative measure the trade-off between agilness and security were proposed by researches such as, for example,

(Keramati and Mirian-Hosseinabadi 2008). These are discussed in the relative section of the research, together with the TISA-XP (Sonia, Singhal, and Banati 2014) and others.

Nonetheless, the dichotomy between the Agile development and Waterfall based security frameworks was not overcome, hinting to an intrinsic problematic associated with hybridizing solutions originating from incompatible ideas.

(Maier, Ma, and Bloem 2017a) proposed once again a security enhanced agile process. The researches this time looked at solutions that would satisfy the ISO "Capability Maturity Model" while retaining a degree of "agilness". Novel to this paper was the introduction of a risk evaluation component, adapted from the OWASP "Application threat modeling". Risk management solutions were also proposed by (Franqueira et al. 2011), (Tigist, Kidane, and Bengt 2013) and (Hutchinson, Maddern, and Wells 2011). These were seen as a effective way to reduce waste and plan ahead which features would require security and to which degree. It should be noted that these risk management models were not meant to be used only on the hybridized solutions derived from the Waterfall methods, but could and were applied also as standalone solutions (Hutchinson, Maddern, and Wells 2011). In particular, (Franqueira et al. 2011) argued that by using "empirical data in the format of public catalogs and the NVD database", the level of expertise required to manage security risks could be reduced. The authors created an iterative risk model consisting of three main phases: risk assessment, treatment and acceptance. At the end of each sprint, risk assessment should provide a guidance to the client and the product owner, guiding their choices on which security feature should be prioritized.

"BugBash" was also designed to tackle testing and quality issues, but was not based on any existing security frameworks. The research presenting it, (Balasubramani et al. 2012), advocated this event as a feasible solution for quality assurance. Lasting from three to five days, "BugBash" was described as a moment in which every member of the development team "turns his/her attention to testing". Teams of "eight to ten" members formed by "a mix of Developers, Product Managers, and Designers" were created. The recommendation was to "have a mixed bag of folks to test the product".

An approach based on the "decomposition of security objectives" presented in paragraph 2.3.6 and firstly shown by the research (Chenxi Wang and Wulf 1997) was the an industrial pilot study by (R. Savola, Frühwirth, and Pietikäinen 2012). In the paper, further discussed in the section relative to the second research question, a risk-driven method to the design of a complex system is discussed. Based on earlier models developed by the researchers, "an iterative methodology for security metrics development" based on the decomposition of security objectives was tested. Moreover, attention was payed to the visualization of these decompositions. The perceived benefit was an increased manageability of the security metrics collected. The tool used for this visualization was the "Metrics Visualization System" (R. M. Savola and Heinonen 2011). This method visualization of visualization was novel in its use of traffic-light color to aide the presentation of data to non-technical development and management members.

### 4.1.3 Tools

This section of the chapter concerns the use of tools suggested by the researches. Some papers presented new ones, created by the researches. Other instead recommended the use of existing one. Overall, 11 papers suggested applications that

matched this group. Like for the other sections, the results will be presented in depth in the following pages.

Reviewing changes for an effective evaluation of the changes in the source code was the focus of the research by (Raschke et al. 2014). The researchers developed a tool, based on open source software, named "Change Detection Analysis". The aim of this instrument was to track every change at the end of the sprint, helping the developers with the security requirements in the ever-changing threat landscape.

Some papers discussed and encouraged the addition of closed-source software, such as "BlackDuck" (Mackey 2018) to perform "Software Composition Analysis", a technique that consists in checking the dependency of the developed software and compare it against known vulnerabilities. Setting aside the goodness of the product, the researcher could not avoid noting a troubling conflict of interest when the authors of the article were also working for the company developing the product.

Other researches did not present tools usable directly to improve the software, but meta-tools to improve the process itself. (Terpstra, Daneva, and Chong Wang 2017) and (Sonia, Singhal, and Banati 2014) followed this approach. (Sonia, Singhal, and Banati 2014) used the integration matrix and weighted matrix from the TISA-XP framework to dynamically select all and only the methods whose agility level was above a certain threshold.

A similar approach was undertaken by (Renatus, Teichmann, and Eichler 2015). The authors in this case presented again a meta-approach, capable of comparing and selecting methods for agile security engineering. Some of the properties that were key in the decision-making of the process were, among others: the choice of additional "Artefacts", "Scrum modifications" to the method, "Security expertise" of the developers. The research thus divided the enhancement aiming at improving security similarly to what this systematic literature review did, albeit by viewing the grouping sets through different lenses.

(D. S. Cruzes et al. 2017) study joined the chorus of researches lamenting "lack of guidelines in practice as well as empirical studies in real-world projects on agile security testing". Testing was considered problematic, since in most companies such activity is executed after development "to detect failures, but typically not to prevent them". What emerged was that "security tests on the system level are to a large extent automated and there is almost no manual security testing". When non-functional requirements were tested, the focus was usually on the performance. Citing (Crispin and Gregory 2009) and the four quadrant of "Agile Testing",

(Munetoh and Yoshioka 2013) Introduced a tool called "RAILROAD", which offered a "model-assisted security testing framework for developing Web applications" by using Ruby-on-Rails code base. The application was designed to abstract the design of the code that was given to it, and took into consideration malicious and improper behaviors. The application inner workings was described in four steps:

1. parse the application code and create a navigation model
2. assess the security design and requirements with the navigation model
3. generate an abuse model by adding the behaviors of the security features, and attack vectors to the navigation model
4. generate the minimum amounts of test cases

No case-study of the application was provided due to its infancy. Further versions would provide additional features in a "plug-in" fashion.

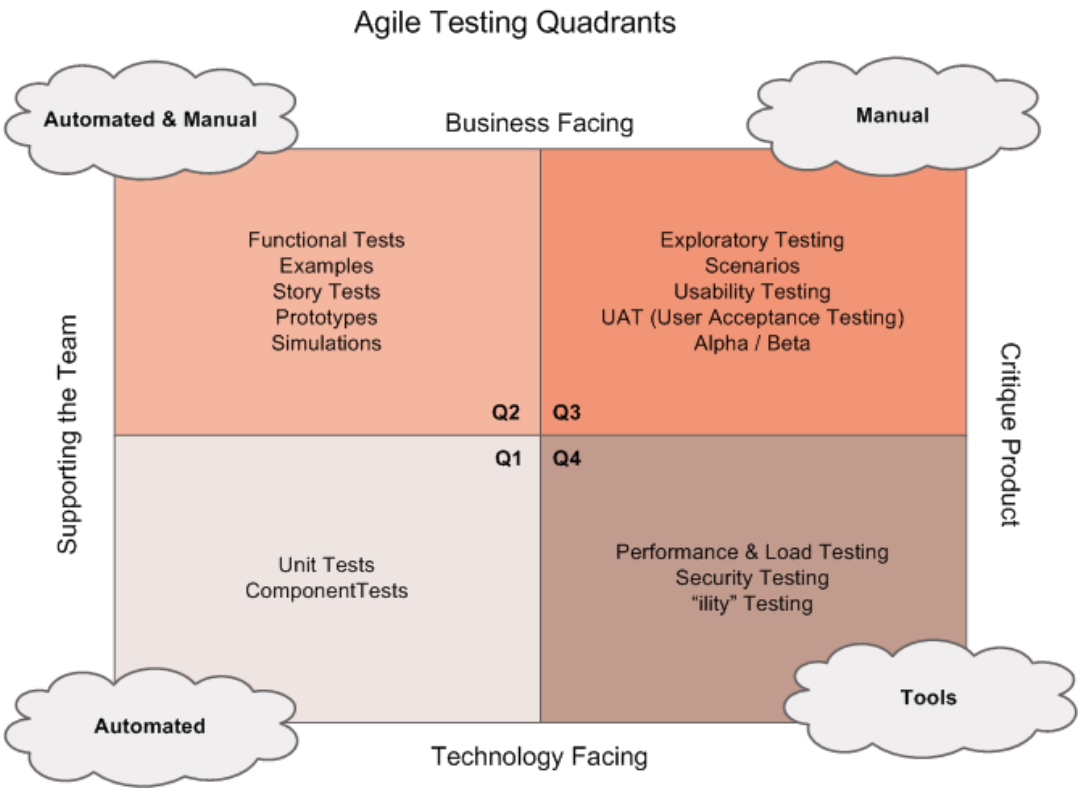


FIGURE 4.2: Agile testing quadrants are widely adopted in practice. By Lisa Crispin, (Crispin and Gregory 2009)

As thoroughly described in paragraph 4.1.1, building secure application using the Agile process is deeply dependent on the developers' personal knowledge of the field. the research by (Felderer and Pekaric 2017) started by acknowledging this fact and looking at possible solutions to expand the knowledge-base that the professionals can access. The framework that the researcher proposed was thus a framework consisting of two components: a "Security Data Collection and Analysis" Component designed to automatically extract informations about known vulnerabilities from public and private sources while the "Security Knowledge Generation" component was tasked with merging and processing the data "in in order to provide it for different roles and various purposes in the agile development process". The production of such an application however proposed several challenges to the researchers. For instance, the identification of quality sources of information was troublesome. Moreover, the vast majority of information gathered was non-relevant (95%). Finally, extracting data from a variegated and non-coherent set of sources was a non trivial task, and for these reasons, the creation of such an automatic "tool/framework" was said to be a formidable objective.

(Hutchinson, Maddern, and Wells 2011) proposed a model to alleviate the complex task of assessing security risks. The conceptual framework was composed by an dataset of vulnerabilities, taken from of online databases and resources that hosted information about known issues (much like the aforementioned (Felderer and Pekaric 2017)), and by a survey directed at the project manager that gauged the level of the "level of risk facing the project".

### 4.1.4 New artifacts for Agile

Not all the solution that affected Agile were borrowed from existing waterfall frameworks and then adapted. Many innovative or re-imagined ideas stemmed from the Agile process itself, from its artifacts and ceremonies. Some of these include: the need to integrate the security requirements into the process artifacts (Bartsch 2011), the “abuse cases” and “evil user stories” (Epstein 2008) and the “security backlog” (Imran Ghani and Jeong 2014). We will analyze them in this last chapter.

In detail, (Barbosa and Barros Sampaio 2015) discussed the introduction of three artifacts and activities to the Agile Process: "Evil User Stories", "Security Backlog" and "Protection Poker", which were considered to be beneficial and to favor the prioritization of security features. "Evil User Stories were described to be documented as normal user stories. An experienced professional, in the role of security master, was tasked to write them in conjunction with the stakeholders, thus prioritizing the protection of the most vital business components. The aim of the "evil stories" was highlighting possible security problems and stimulating the development of countermeasures. The "Security Backlog" was described as a list of "features in the Product Backlog that need security attention". Again, the task of writing the "security backlog" was assigned to the security master figure. The benefit of this additional artifact was an ease in the "traceability of security requirements". Finally, the "protection poker" activity was discussed. It occurred during the "planning poker", and it concerned the security requirements. It was supposed to develop and extend the discussion on security issues. The higher the value of the estimate, the more attention was to be given to the security risk.
The work by (Imran Ghani and Jeong 2014) reiterates the addition of a "security backlog" and the role of the "security master" in charge of it. Novel to this approach, some of the artifacts of Scrum (product, release and sprint backlogs) were taken and then staked against the security principles by (Saltzer and Schroeder 1975). The "security backlog" component was here described as a "step" in which features pass before reaching the actual product backlog. To validate this approach, the authors analytically studied the degree of agility of each phase of the process using the 4-dat framework. The findings indicated that the degree of agility improved, according to their model, after the addition of the "security backlog", but no empirical validation was performed. The case-study performed by (Rindell, Hyrynsalmi, and Leppänen 2016) adopted a "lightweight" version of the methods discussed above. In this case, the researcher introduced the security activities directly into the product backlog, and not adding any further artifact to the Scrum process. An innovative way to build security inside Agile was investigated by (Othmane et al. 2014) and was based on the "security assurance cases" presented by (Weinstock, Lipson, and Goodenough 2012). The most common method to implement security assurance was the use of checklists to verify security requirements. These cases were instead described as a "a semi-formal approach" to it and were described to be composed by a set of claims, arguments and evidences. To efficiently structure the cases, tree-diagrams were employed. The root of each tree was a main security claim, the nodes were more descriptive and detailed focused sub-claims, and its leafs the evidences. The novel model used secure engineering activities such as risk assessment, engineering, and security assurance in conjunction with "security assurance cases". Focus of this approach was ensuring compatibility with the iterative nature of the Agile development process. Although a case-study performed by the researchers, no third party or empirical evaluation of the method was performed. Moreover, the scalability of this

method was highlighted by the researchers as a possible issue, since tracking the impact of new user stories to the security assurance cases became harder and harder. The effort required grew "very fast" by each component, claim, evidences and arguments added. (A. F. B. Arbain, Ghani, and Kadir 2014) proposed a brand new approach creating a "Traceability Process Model"(TPM) that takes non-functional requirements into consideration. The NFR are considered critical to the quality of any project being developed and include "security" and "performance" of the system. The new TPM was produced to be agile-compatible, after noting that none of the existing models was designed with this capability in mind. Moreover, the researchers identified specific problems arising from applying standard traceability models to Agile, in particular "propagation issue" and "consistency issue". Further industry case study where also planned to further validate this model, due to the lack of empirical data.

## 4.2 RQ2: Which of the solutions discussed have performed best in pilot studies?

Only 7 papers included in this systematic literature review included real world data and were suitable to answer this research question. The researcher thought that this question could help practitioners in two different ways. First, it would provide them with tested solutions that have been proven beneficial in real world applications. Second of all, it would suggests which issues and difficulties other developers and companies had faced when adopting these techniques. Thus, the two research questions provide the same answers, but view them under two very different lenses. Moreover this section highlights the staggering lack of empirical data in the field and suggests possible future work for other researchers.

| Fully successful | Mixed results | Negative |
|---|---|---|
| (Baca, Boldt, et al. 2015) | (Terpstra, Daneva, and Chong Wang 2017) | (Poller et al. 2017) |
| (Choliz, Vilas, and Moreira 2015) | (Oyetoyan, D. Cruzes, and Jaatun 2016) | |
| | (R. Savola, Frühwirth, and Pietikäinen 2012) | |
| | (Rindell, Hyrynsalmi, and Leppänen 2016) | |

The solutions that the companies had adopted strove to be cost-effective, since the projects were always constrained by budget and profitability.

### 4.2.1 External consultants and workshops

While providing a growth in the interest of developers and their knowledge in the field, security training in some cases failed to produce effective changes in the development cycle. It is the case emphasized in (Poller et al. 2017) where developers were not assured that the training actually improved significantly the quality of their software. They instead highlighted how changes in the management of the organization must take place to fully take advantage of the improved know-how of the developers.

It should be noted that (Terpstra, Daneva, and Chong Wang 2017) results, which were extrapolated from publicly available online resources, found that "practitioners did not search for complex and mathematics-grounded techniques. Instead they applied simple practices with a focus on incorporating those in the social environment of their projects". (Oyetoyan, D. Cruzes, and Jaatun 2016) argued that secure design training was the most demanded and sought after coaching need, but further studies were required to validate such statement.

### 4.2.2 Addition of professional figures to the development team

(Baca, Boldt, et al. 2015) reported a success in integrating a security enhanced process, called SEAP, with Agile. The company involved in the study was "Ericsson" and the development team was composed, for this particular project, by 88 staff members, distributed among 8 development teams. The approach taken relied on risk analysis and on 4 additional professional figures dedicated to security, which are described in the paragraph 4.1.1. The results showed that the usage of SEAP achieved a great reduction of issues in the code thus improving security:

- The proportion of risks that were corrected within the software version in development increased more than five times
- The number of unhandled risks decreased significantly
- More severe risks were found because of a more detailed risk analysis
- The time to solve security issues was greatly reduced

On the other hand, the cost payed to achieve this improvement came from the increased spending on the 4 additional professionals per team. In particular:

- The daily security work done by all security resources proved essential, the risk analysis composed only a fraction of the effort
- The cost dedicated to solve security issues went from the 1% estimate to 5% of the total cost project.

The research argued that a cost-comparison with the non-enhanced process would be nonsensical, since it would not be plausible to calculate the direct-monetary gains and costs of it.

### 4.2.3 Frameworks hybridization

(R. Savola, Frühwirth, and Pietikäinen 2012) performed as well a pilot study at "Ericsson", an experience whose results were mixed. In this case the Agile process had been modified by the addition of a "risk-driven" security design, and by collecting information through a "hierarchical metric" system, based on the decomposition of security objectives, which enabled the developers to better connect the SO with the corresponding metrics. Measuring these security metrics was viewed as positive by the practitioners who emphasized the improved ability of compliance with the standards, as well as "bird-view" vision of the security throughout the project.

In the study by (Choliz, Vilas, and Moreira 2015), the synchronization between the the development team using Agile and the security needs was studied and proved successful. The company name was not disclosed, but it was said to develop software products and "software-as-a-service" solutions. The the security and the development team were kept separate in this process. The efforts to align the two teams

focused on applying Microsoft SDL practices (every-sprint, bucket one-time) and automated and semi-automated tools to aide the testing. The firm tested the new process on two projects, and compared the findings with other two projects, developed this time with a waterfall process. The results showed that issue-detection happened much earlier on the Agile process projects. No improvements in the number of security issues discovered nor in terms of cost were found in this particular case.

### 4.2.4 Addition of artifacts to Agile

(Rindell, Hyrynsalmi, and Leppänen 2016) adopted only a slightly modified version of Scrum, enhancing the process with weekly refinements to the product backlog. The final product was a secure identity management system and its management processes which had to be compliant with the "VAHTI" security instructions, a set of guidelines published by the Ministry of Finance of the Finnish government. The outcome was positive but the process showed mixed results. The study highlighted how this approach to security, which is, adding the security concerns inside the product backlog as items to develop, was sufficient to satisfy the "VAHTI" requirements, but may have not been cost-effective. it was noted that Scrum may have not been used fully, since many security activities were performed in "spikes" of work, outside the sprints. Thus an accurate assessment of this methodology could not be performed.

# Chapter 5

# Discussion

In this section, the solutions presented in depth in the findings chapter are discussed. Particular focus was given in comparing the research questions findings when these differed. Subsequently, the limitation of this study and future research are discussed.

TABLE 5.1: Suggested actions for implementing security in Agile

| Solutions proposed | Sources | Number of cases |
|---|---|---|
| **Solutions addressing the artifacts** | | 7 (18%) |
| Addition of security related artifacts | (Maier, Ma, and Bloem 2017b), (Imran Ghani and Jeong 2014), (Bartsch 2011),(Barbosa and Barros Sampaio 2015), (Epstein 2008) | 5 |
| Enhancement of existing artifacts | (Rindell, Hyrynsalmi, and Leppänen 2016) | 1 |
| Risk-analysis artifacts | (Baca, Boldt, et al. 2015), | 1 |
| **Human Factor** | | 17 (44%) |
| Consultants and training | (Poller et al. 2017),(Caldwell 2015), (Terpstra, Daneva, and Chong Wang 2017), (Oyetoyan, D. Cruzes, and Jaatun 2016), (Camacho, Marczak, and D. Cruzes 2016), (Bartsch 2011), (D. S. Cruzes et al. 2017), (Adelyar and Norta 2017), (Tigist, Kidane, and Bengt 2013), (Barbosa and Barros Sampaio 2015), (Felderer and Pekaric 2017) | 11 |
| Addition of a security dedicated team member | (Imran Ghani and Jeong 2014), (Epstein 2008), (Alnatheer et al. 2013) | 3 |
| Addition of a full security team | (Baca, Boldt, et al. 2015),(Choliz, Vilas, and Moreira 2015) | 2 |
| Improvements of the workplace | (Loser and Degeling 2014) | 1 |

| Solutions proposed | Sources | Number of cases |
|---|---|---|
| **Solutions addressing the Agile process** itself | | 18 (47%) |
| hybridize waterfall security frameworks | (Maier, Ma, and Bloem 2017b), (Baca and Carlsson 2011), (Choliz, Vilas, and Moreira 2015), (Rindell, Hyrynsalmi, and Leppänen 2015), (Siponen, Baskerville, and Kuivalainen 2012), (Tigist, Kidane, and Bengt 2013), (Othmane et al. 2014), (Rindell, Hyrynsalmi, and Leppänen 2016), (Renatus, Teichmann, and Eichler 2015) | 9 |
| Agile enhanced with risk-evaluation | (Maier, Ma, and Bloem 2017a), (Franqueira et al. 2011), (Hutchinson, Maddern, and Wells 2011), (Baca, Boldt, et al. 2015) | 4 |
| Dynamic frameworks based on "agilness" | (Keramati and Mirian-Hosseinabadi 2008), (Sonia, Singhal, and Banati 2014) | 2 |
| Decomposition and traceability of SO | (R. Savola, Frühwirth, and Pietikäinen 2012),(A. F. B. Arbain, Ghani, and Kadir 2014) | 2 |
| Specific testing activity | (Balasubramani et al. 2012) | 1 |
| **Solutions addressing the tools** | | 11 (26%) |
| Agile-process-evaluation tools | (Terpstra, Daneva, and Chong Wang 2017), (Kumar et al. 2012), (Sonia, Singhal, and Banati 2014), (Renatus, Teichmann, and Eichler 2015), (Bansal and Jolly 2014) | 5 |
| Automatic testing tools | (Munetoh and Yoshioka 2013) (D. S. Cruzes et al. 2017) | 2 |
| Changes evaluation software | (Raschke et al. 2014) | 1 |
| Closed source software | (Mackey 2018) | 1 |
| Vulnerability gathering tools | (Felderer and Pekaric 2017) | 1 |
| Risk-analysis tools | (Hutchinson, Maddern, and Wells 2011) | 1 |

## 5.1 Observations on the proposed solutions

As highlighted by the table, the use of consultants and workshops to perform security training directed to the development team, was the suggestion with the highest consensus among the research papers. The cause of this agreement may originate from established practices. Security training and awareness is in fact a prerequisite to an established framework such Microsoft SDL. Incidentally, this practice did not

work well in the industry-study by (Poller et al. 2017). The practitioners did see an improvement in their awareness and perception of security, but were concerned that without changes in the way the work was being done, the benefit would have been minimal. It thus seems that training may be a necessary but not sufficient driver for the implementation of security.

The addition of specialized figures or teams dedicated to security echoed established business practices, and proved successful in the industry (Baca, Boldt, et al. 2015). Nonetheless, this practice seems to be in stark contrast with the Agile "cross-functional" teams mantra. Moreover, this methodology was not proved (nor disproved) to be cost-effective. It undoubtedly does prove useful for those practitioners having to develop application with strict security requirements.

Mixing established security frameworks with Agile was the second most popular solution. The most recurring frameworks were: Microsoft SDL, Cigatel Touchpoints, CLASP principles by OWASP and the Capability Maturity Model. While the quality of these frameworks is well established, some issues arose when these were hybridized with Agile, mostly due to their focus on a sizable design phase. The results of the pilot-studies showed in general positive results (R. Savola, Frühwirth, and Pietikäinen 2012) and (Choliz, Vilas, and Moreira 2015), but did not test the proposed solutions presented in this study exhaustively. Further in depth testing will be required if a comparison between the methods is to be performed. Recognizing the difficulties of integrating frameworks designed for waterfall to Agile, many studies presented meta-tools to calculate the degree of "agilness" of the security activities. These are listed in the table 5.1 as "Agile-process evaluation tools". Some of these instruments, such as for instance (Sonia, Singhal, and Banati 2014), allowed the practitioners to select dynamically the activities above a certain "agility threshold" to best adapt to the security requirements required. Such tools, if perfected, may prove incredibly valuable for a software company. They provide an easily tunable method to choose security activities which does not rely on the developers' knowhow. It would also be highly appreciated by the management, providing a baseline idea of the costs required to achieve a certain software trustworthiness level.

Finally, a number of additional artifacts stemming from Agile were proposed. Some, like a the "security backlog" or the "evil user stories" were totally dedicated to the enhancement of the security. Other researches preferred lightweight methods, such as embedding the security requirements in the product backlog. Whichever the approach, a standardization and refinement of these additional artifacts from one of the main Agile processes may prove constructive. Because of their resemblance with Agile artifacts already in use, these solutions be simple to include in the processes in use at any company that has adopted Agile.

## 5.2 Limitation of this study

Every systematic literature review would be better conduced by two or more researchers, to avoid personal bias. This is especially true in the planning and conducting phases. The design of the research questions and the selection of the research papers, in particular, should be conduced independently by each researcher, and then the results should be confronted to establish a consensus. Due to the fact that this SLR was a bachelor thesis, the researcher could not perform this independent check, being the only author of the study. Moreover, every systematic literature review is affected by the publication bias, since its findings are only based on researches published in the field (Kitchenham 2004).

## 5.3 Related work

A systematic literature review with a similar focus but on a smaller scale has been conducted by (Rindell, Hyrynsalmi, and Leppänen 2017), many beneficial security activities for Agile are discussed. Some other systematic literature review published on the Agile-Security topic were (Tigist Eyader Ayalew 2012) (Master thesis) and (A. Arbain, Ghani, and Jeong 2014). A tertiary study collecting a group of systematic literature review on the Agile software development was also produced by (Hoda et al. 2017).

## 5.4 Future research

Due to the ubiquity of Agile software development methods and to the growing necessity for secure software, the study of security focused Agile processes is only going to get more important. In this systematic literature review, the researcher highlighted the endemic lack of empirical data. Thus, future works should be directed to pilot-studies to validate the theoretical solutions proposed by the academia.

Moreover, the creation of tools to precisely evaluate the costs and benefits of enhancing security in software products could highly increase the cost-effectiveness of the solutions.

Finally, the studies had shown a high degree of dependability on the developers knowhow when writing secure software. Methods to decrease this reliability on the practitioners knowledge, such as easily accessible vulnerability-databases, should be investigated

# Chapter 6

# Conclusions

In this thesis, the researcher presented the results of a systematic literature review that inquired which methods were the most effective to implement security within the Agile software development process.

The research analyzed the results from 39 research papers published between 2011 and 2018, and coded the suggestions into four different groups: additional artifacts in Agile, human factor, addition to the Agile process and tools.

The first research question asked which solutions best addressed the problem of creating secure software in the Agile process. The answer that shown the highest consensus among the academic community was to improve the "human factor" by means of training. Additions to the development team were also discussed in their drawbacks and benefits. Subsequently, frameworks and to the Agile process were suggested, mostly by hybridizing waterfall security development techniques with Agile. Tools to measure the degree of agilness and to increase the security awareness of the developers were then proposed. Finally, the articles that belonged to the "artifacts for agile" group suggested new additions such as the "security backlog" or the "evil user stories".

Extrapolating meaningful results to answer the second research question, which asked which solutions performed best in the industry and pilot studies, resulted challenging due to the lack of data. Only seven papers presented empirical evidences, suggesting that further research is needed to validate most of the solutions presented. Nonetheless, the results of the second research question shown mostly positive or partially positive results. Evaluating the cost-effectiveness of the methods, thus providing an easy way to compare them, proved difficult.

For future research, the author suggests performing pilot and case studies to increase the body of empirical knowledge, together with methods to evaluate the cost-effectiveness of the solutions and to decrease the reliability on the practitioners' know-how.

# Bibliography


Adelyar, Sayed Hassan and Alex Norta (2017). "Security benefits for agile software development". In: *2017 7th International Workshop on Computer Science and Engineering, WCSE 2017*. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85027882539&partnerID=40.

Alnatheer, Ahmed et al. (2013). "Agile Security Methods: an Empirical Investigation". In: *Proceedings of the IASTED International Conference, Software Engineering*. DOI: 10.2316/P.2014.810-011.

Arbain, Adila Firdaus Binti, Imran Ghani, and Wan Mohd Nasir Wan Kadir (2014). "Agile non functional requiremnents (NFR) traceability metamodel". In: *2014 8th. Malaysian Software Engineering Conference (MySEC)*, pp. 228–233. DOI: 10.1109/MySec.2014.6986019.

Arbain, Adila, Imran Ghani, and Seung Ryul Jeong (2014). "A Systematic Literature Review on Secure Software Development using Feature Driven Development (FDD) Agile Model". In: 15. DOI: 10.7472/jksii.2014.15.1.13.

Baca, Dejan, Martin Boldt, et al. (2015). "A Novel Security-Enhanced Agile Software Development Process Applied in an Industrial Setting". In: *2015 10th International Conference on Availability, Reliability and Security*. DOI: 10.1109/ARES.2015.45.

Baca, Dejan and Bengt Carlsson (2011). "Agile development with security engineering activities". In: pp. 149–158. DOI: 10.1145/1987875.1987900.

Balasubramani, Uma M. et al. (2012). "Bug Bash: An Efficient Approach to Increase Test Coverage and Ensure Product Quality in an Agile Environment". In: *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. ISSN: 1097-5659;10975659. DOI: 10.1109/RADAR.2011.5960496. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5960496.

Bansal, Sushil Kumar and Ashish Jolly (2014). "An encyclopedic approach for realization of security activities with agile methodologies". In: *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, pp. 767–772. DOI: 10.1109/CONFLUENCE.2014.6949242.

Barbosa, Dayanne Araujo and Suzana Cândido de Barros Sampaio (2015). "Guide to the Support for the Enhancement of Security Measures in Agile Projects". In: *2015 6th Brazilian Workshop on Agile Methods (WBMA)*, pp. 25–31. DOI: 10.1109/WBMA.2015.9.

Bartsch, Steffen (2011). "Practitioners' Perspectives on Security in Agile Development". In: *2011 Sixth International Conference on Availability, Reliability and Security*, pp. 479–484. DOI: 10.1109/ARES.2011.82.

Basili, Victor R., Richard W. Selby, and David H. Hutchens (1986). "Experimentation in software engineering". In: *IEEE Transactions on Software Engineering* SE-12, pp. 733–743. DOI: 10.1109/TSE.1986.6312975.

Beck, Kent et al. (2001). *Manifesto for Agile Software Development*. URL: www.agilemanifesto.org.

Bishop, Matt (2003). "What Is Computer Security?" In: *IEEE Security And Privacy* 1, pp. 67–69. DOI: 10.1109/MSECP.2003.1176998.

Brooks, Richard R. (2013). *Introduction to Computer and Network Security: Navigating Shades of Gray*. Chapman and Hall/CRC. ISBN: 9781439860717.

Caldwell, Tracey (2015). "Taking agile development beyond software - What are the security risks?" In: *Network Security*. DOI: 10.1016/S1353-4858(15)30110-0.

Camacho, Cristina Rosa, Sabrina Marczak, and Daniela Cruzes (2016). "Agile Team Members Perceptions on Non-functional Testing: Influencing Factors from an Empirical Study". In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pp. 582–589. DOI: 10.1109/ARES.2016.98.

Choliz, Jesus, Julian Vilas, and Jose Moreira (2015). "Independent Security Testing on Agile Software Development: A Case Study in a Software Company". In: *2015 10th International Conference on Availability, Reliability and Security*, pp. 522–531. DOI: 10.1109/ARES.2015.79.

*CLASP principles*. URL: https://www.owasp.org/index.php/CLASP_Security_Principles.

Cohn, Mike (2009). *succeeding with agile software development using scrum*. Addison Wesley.

Collabnet (2018). *12th Annual State of Agile Report, April 9, 2018*. URL: https://www.collab.net/news/press/collabnet-versionone-announces-12th-annual-state-agile-report.

Crispin, Lisa and Janet Gregory (2009). *Agile Testing: A Practical Guide for Testers and Agile Teams*. ISBN: 9780321534460.

Cruzes, Daniela Soares et al. (2017). "How is Security Testing Done in Agile Teams? A Cross-Case Analysis of Four Software Teams". In: *Lecture Notes in Business Information Processing* 283, pp. 201–216. DOI: 10.1007/978-3-319-57633-6_13.

d'Agapeyeff, Alexander (1969). *Software Engineering, Report on a conference sponsored by the NATO Science Committee*, pp. 15–15. URL: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF.

Dimensional research (2018). *Testing trends for 2018: a survey of development and testing professionals*. 355 W Olive Ave, Sunnyvale, CA 94086, USA. URL: https://cdn.agilitycms.com/sauce-labs/white-papers/sauce-labs-state-of-testing-2018.pdf.

Epstein, Richard G. (2008). "Getting Students to Think About How Agile Processes can be Made More Secure". In: *2008 21st Conference on Software Engineering Education and Training*, pp. 51–58. DOI: 10.1109/CSEET.2008.13.

Felderer, Michael and Irdin Pekaric (2017). "Research Challenges in Empowering Agile Teams with Security Knowledge Based on Public and Private Information Sources". In: *SecSE@ESORICS*.

Franqueira, Virginia N. L. et al. (2011). "Towards agile security risk management in RE and beyond". In: *Workshop on Empirical Requirements Engineering (EmpiRE 2011)*, pp. 33–36. DOI: 10.1109/EmpiRE.2011.6046253.

Hoda, Rashina et al. (2017). "Systematic literature reviews in agile software development: A tertiary study". In: 85. DOI: 10.1016/j.infsof.2017.01.007.

Humphrey, Watts S. (1988). "Characterizing the software process: a maturity framework". In: *IEEE Software* 5 (2), pp. 73–79. DOI: 10.1109/52.2014.

Hutchinson, Damien, Heath Maddern, and Jason Wells (2011). "An agile it security model for project risk assessment". In: *Proceedings of the 9th Australian Information Security Management Conference)*.

Imran Ghani, Zulkarnain Azham and Seung Ryul Jeong (2014). "Integrating Software Security into Agile-Scrum Method". In: *KSII Transactions on Internet and Information Systems* 8. DOI: 10.3837/tiis.2014.02.019. URL: http://www.itiis.org/digital-library/manuscript/688.

Keblawi, Feisal and Dick Sullivan (2006). "applying the Common Criteria in Systems Engineering". In: *IEEE Security and Privacy* 4, pp. 50–55. DOI: 10.1109/MSP.2006.35.

Keramati, Hossein and Seyed-Hassan Mirian-Hosseinabadi (2008). "Integrating software development security activities with agile methodologies". In: *2008 IEEE/ACS International Conference on Computer Systems and Applications*, pp. 749–754. DOI: 10.1109/AICCSA.2008.4493611.

Kitchenham, Barbara (2004). "Procedures for Performing Systematic Reviews". In: ISSN: 1353-7776. URL: http://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf.

Kumar, Upendra et al. (2012). "Dependable solutions design by Agile Modeled Layered Security Architectures". In: *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*. DOI: 10.1007/978-3-642-27299-8_53. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84888412062&doi=10.1007%2f978-3-642-27299-8_53&partnerID=40&md5=c19109e41c28e2e80105e948fe7e6973.

Loser, Kai-Uwe and Martin Degeling (2014). "Security and Privacy as Hygiene Factors of Developer Behavior in Small and Agile Teams". In: *HCC*. DOI: 10.1007/978-3-662-44208-1_21.

Mackey, Tim (2018). "Building open source security into agile application builds". In: *Network Security*. ISSN: 1353-4858. DOI: 10.1016/S1353-4858(18)30032-1.

Maier, Patrik, Zhendong Ma, and Roderick Bloem (2017a). "Towards a Secure SCRUM Process for Agile Web Application Development". In: DOI: 10.1145/3098954.3103171.

— (2017b). "Towards a Secure SCRUM Process for Agile Web Application Development". In: *Proceedings of the 12th International Conference on Availability, Reliability and Security* 5. ISSN: 1084-6654. DOI: 10.1145/3098954.3103171.

McGraw, Gary (2005). "A portal for software security". In: *IEEE Security and Privacy*. DOI: 10.1109/MSP.2005.88.

— (2006). *Software Security, building security in*. IEEE. ISBN: 0-7695-2684-5. DOI: 10.1109/ISSRE.2006.43.

*Microsoft SDL*. URL: https://www.microsoft.com/en-us/sdl.

*Microsoft SDL for Agile*. URL: https://www.microsoft.com/en-us/sdl/discover/sdlagile.aspx.

Munetoh, Seiji and Nobukazu Yoshioka (2013). "RAILROADMAP: An agile security testing framework for web-application development". In: *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013* 5. ISSN: 1084-6654. DOI: 10.1109/ICST.2013.80. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84883372117&doi=10.1109%2fICST.2013.80&partnerID=40&md5=cef7b624f16548716a708d94a32126b4.

Othmane, Lotfi Ben et al. (2014). "Extending the Agile Development Process to Develop Acceptably Secure Software". In: *IEEE Transactions on Dependable and Secure Computing* 11, pp. 497–509. DOI: 10.1109/TDSC.2014.2298011.

*OWASP dependency check*. URL: www.owasp.org/index.php/OWASP_Dependency_Check.

*OWASP top 10* (2017). URL: www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf.

Oyetoyan, Tosin Daniel, Daniela Cruzes, and Martin Gilje Jaatun (2016). "An Empirical Study on the Relationship between Software Security Skills, Usage and Training Needs in Agile Settings". In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pp. 548–555. DOI: 10.1109/ARES.2016.103.

Poller, Andreas et al. (2017). "Can Security Become a Routine?: A Study of Organizational Change in an Agile Software Development Group". In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. DOI: 10.1145/2998181.2998191.

Project Management Institute (2017). *Pulse of the Profession*. 14 Campus Boulevard Newtown Square, PA 19073-3299 USA. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.

Raschke, Wolfgang et al. (2014). "Supporting evolving security models for an agile security evaluation". In: *2014 IEEE 1st International Workshop on Evolving Security and Privacy Requirements Engineering, ESPRE 2014 Proceedings*. DOI: 10.1109/ESPRE.2014.6890525.

Renatus, Stephan, Clemens Teichmann, and Jörn Eichler (2015). "Method Selection and Tailoring for Agile Threat Assessment and Mitigation". In: *2015 10th International Conference on Availability, Reliability and Security*, pp. 548–555. DOI: 10.1109/ARES.2015.96.

Rindell, Kalle, Sami Hyrynsalmi, and Ville Leppänen (2015). "A comparison of security assurance support of agile software development methods". In: *CompSysTech*. DOI: 10.1145/2812428.2812431.

— (2016). "Case Study of Security Development in an Agile Environment: Building Identity Management for a Government Agency". In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pp. 556–563. DOI: 10.1109/ARES.2016.45.

— (2017). "Busting a Myth: Review of Agile Security Engineering Methods". In: *ARES*. DOI: 10.1145/3098954.3103170.

Rosenberg, Doug and Matt Stephens (2003). "Extreme programming refactored: the case against XP". In:

Royce, Winston W. (1970). "Managing the development of large software systmes". In: IEEE, pp. 328–388. URL: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf.

Saltzer, Jerome Howard and Michael D. Schroeder (1975). "The protection of information in computer systems". In: *Proceedings of the IEEE* 63.9, pp. 1278–1308. ISSN: 0018-9219. DOI: 10.1109/PROC.1975.9939.

Savola, Reijo M. and Petri Heinonen (2011). "A Visualization and Modeling Tool for Security Metrics and Measurements Management". In: *2011 Information Security for South Africa*. DOI: 10.1109/ISSA.2011.6027518.

Savola, Reijo, Christian Frühwirth, and Ari Pietikäinen (2012). "Risk-Driven Security Metrics in Agile Software Development - An Industrial Pilot Study". In: *J. UCS* 18, pp. 1679–1702. DOI: 10.3217/jucs-018-12-1679.

Schwaber, Ken and Mike Beedle (2002). *Agile Software Development with Scrum*. PTR Upper Saddle River, NJ, USA: 1st Prentice Hall. ISBN: 0130676349.

Schwaber, Ken and Jeff Sutherland (2017). *The Scrum Guide*. URL: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf.

Siponen, Mikko T., Richard Baskerville, and Tapio Kuivalainen (2012). "Integrating Security into Agile Development Methods". In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 185a–185a. DOI: 10.1109/HICSS.2005.329.

Smith, Richard E. (2012). "A Contemporary Look at Saltzer and Schroeder's 1975 Design Principles". In: *IEEE Security and Privacy*. DOI: 10.1109/MSP.2012.85.

Sonia, Archana Singhal, and Hema Banati (2014). "FISA-XP: an agile-based integration of security activities with extreme programming". In: *ACM SIGSOFT Software Engineering Notes* 39, pp. 1–14. DOI: 10.1145/2597716.2597728.

Terpstra, Evenynke, Maya Daneva, and Chong Wang (2017). "Agile practitioners' understanding of security requirements: Insights from a grounded theory analysis". In: *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017*. DOI: 10.1109/REW.2017.54. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85034647916&doi=10.1109%2fREW.2017.54&partnerID=40&md5=d5eb8e38f87efcca8dd03f97ab148ae8.

*The OWASP foundation*. URL: www.owasp.org.

Tigist Eyader Ayalew, Tigist Abreham Kidane (2012). "Identification and Evaluation of Security Activities in Agile Projects. A Systematic Literature Review and Survey Study". SE - 371 79 Karlskrona Sweden: School of Computing, Blekinge Institute of Technology.

Tigist, Ayalew, Tigist Kidane, and Carlsson Bengt (2013). "Identification and evaluation of security activities in agile projects". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8208 LNCS, pp. 139, 153. DOI: 10.1007/978-3-642-41488-6_10. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84890878492&doi=10.1007%2f978-3-642-41488-6_10&partnerID=40&md5=ea75f9c1506d7d4d5f9629d0772b344f.

Wang, Chenxi and William A. Wulf (1997). "Towards a framework for security measurement". In: pp. 522–533.

Weinstock, Charles B., Howard F. Lipson, and John B. Goodenough (2012). "Arguing Security – Creating Security Assurance Cases". In: URL: https://resources.sei.cmu.edu/asset_files/WhitePaper/2013_019_001_293637.pdf.

Williams, Jeff (2014). *The Unfortunate Reality of Insecure Libraries*. URL: https://www.owasp.org/images/7/70/ASDC12-The_Unfortunate_Reality_of_Insecure_Libraries.pdf.