



Towards a Conceptual Framework for Security Requirements Work in Agile Software Development

Inger Anne Tøndel, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

 <https://orcid.org/0000-0001-7599-0342>

Martin Gilje Jaatun, SINTEF Digital, Oslo, Norway

 <https://orcid.org/0000-0001-7127-6694>

ABSTRACT

Security requirement work plays a key role in achieving cost-effective and adequate security in a software development project. Knowledge about software companies' experiences of security requirement work is important in order to bridge the observed gap between software security practices and security risks in many projects today. Particularly, such knowledge can help researchers improve on available practices and recommendations. This article uses the results of published empirical studies on security requirement work to create a conceptual framework that shows key concepts related to work context, this work itself and the effects of this work. The resulting framework points to the following research challenges: 1) Identifying and understanding factors important for the effect of security requirements work; 2) Understanding what is the importance of the chosen requirements approach itself, and; 3) Properly taking into account contextual factors, especially factors related to individuals and interactions, in planning and analysis of empirical studies on security requirements work.

KEYWORDS

Agile Software Development, Conceptual Framework, Empirical Studies, Literature Review, Security Requirements, Software Security

INTRODUCTION

In today's interconnected world, we would claim that software security is an aspect to consider in most software development projects. Currently, agile development methodologies are prominent in software development. Such methods are used even for large scale development (Dikert, Paasivaara, & Lassenius, 2016) and for security critical and safety critical software (Hanssen, Stålhane, & Myklebust, 2018; Heeager & Nielsen, 2018; Oueslati, Rahman, & ben Othmane, 2015). Thus, good ways of working with security within an agile development paradigm is necessary.

There has been done a lot of work on suggesting ways to deal with software security in agile development projects, including proposals for integrating security into agile methodologies like XP

DOI: 10.4018/IJSSSP.2020010103

(Aydal, Paige, Chivers, & Brooke, 2006) and Scrum (Pohl & Hof, 2015). In 2009 the Microsoft Security Development Lifecycle (SDL) (Howard & Lipner, 2006; Microsoft, n.d.) was released in a version specific for agile development (Agile SDL) (Microsoft, 2009), and abuser stories have for some time been a suggested way of representing security requirements within agile development (Peeters, 2005). Additionally, there exist method-agnostic approaches to software security that should be possible to integrate with agile development, such as the touchpoints for software security (McGraw, 2004, 2006), the Building Security in Maturity Model (BSIMM) (McGraw, Migues, & West, 2018) and the OWASP Software Assurance Maturity Model (SAMM) (OWASP, n.d.). There thus seems to be no lack of methods for doing software security work also within an agile paradigm. Still, many have observed that security is frequently not given proper attention in software development projects today (Tøndel, Jaatun, Cruzes, & Moe, 2017; Nicolaysen, Sassoon, Line, & Jaatun, 2010; Terpstra, Daneva, & Wang, 2017).

As is well communicated by the abovementioned software security approaches, software security should be an integrated part of development and have a role in the various software development activities, including requirements, design, coding, testing, deployment and operations. Security is not something that can be successfully added on as an afterthought, but should be built into the system from the beginning. This however means that the total number of suggested security activities can be quite overwhelming. It is possible for projects to spend a lot of effort on security, even over-spending, if not properly addressing the security needs. Thus, we consider security requirements work as foundational to achieving cost-effective security in a project.

In this article, we define software security requirements work as activities performed in relation to a software development project to: 1) decide whether and how to identify security needs, risks or requirements for a project; 2) do the requirements elicitation; 3) communicate the identified security needs, risks or requirements, and; 4) integrate these and make priorities related to them in development. By agile development we mean software development that in large part is guided by the Agile principles, as outlined in the Agile Manifesto (Beck et al., 2001), including various methods such as Scrum and XP. Compared to a waterfall development approach, requirements management in agile development is “far more temporal, interactive and just in time” (Leffingwell, 2010). Additionally, the need for requirements prioritization can be considered to be built into the approach; “[w]e admit up front that we can’t implement (nor even discover) all potential requirements” (Leffingwell, 2010). Security is only one of the types of requirements a development project needs to consider. When negotiating the three variables cost, schedule and requirements (Leffingwell, 2010), requirements may be modified or dropped altogether.

There exist few empirical studies on how security requirements are handled in software development projects (Terpstra et al., 2017), thus “[h]ow practitioners in the field think about security requirements and how they devise their processes of coping with the issues these requirements pose, is hardly known” (Terpstra et al., 2017). Empirical studies of software security practices within agile development can help us understand what makes companies and projects adopt (or not adopt) software security practices, how different practices may help, what works well and what is challenging in certain contexts, etc. Such knowledge is important in order to bridge the observed gap between software security practices and software security needs and risks in many projects today. In particular, such knowledge can help researchers improve on existing support provided to agile development projects in terms of available practices and recommendations.

In this article, we aim to improve understanding of security requirements work within an agile development approach. Our study aims to answer the following research questions:

- What factors are found in current empirical research to influence and/or characterize security requirements work in agile projects in an industry setting?
- How do these factors impact the security requirements work and its effect?

We build on published empirical studies that cover security requirements work, emphasizing results from studies performed in an industry setting. The findings from these studies are used to build a conceptual framework that shows important concepts that impact and/or characterize security requirements work, and the relations between these concepts. The conceptual framework is a step towards a comprehensive understanding of security requirements work in agile projects, and can act as a basis for further research on this topic, both in prioritizing which research studies to undertake, and as input to planning and analysis in future empirical studies within this topic.

The remainder of this article is structured as follows. It starts by giving an overview of literature reviews related to security requirements work in agile development and introducing two previously suggested conceptual framework related to this type of work. Then it moves on to explaining the method used to identify empirical sources and construct the conceptual framework. Following the method description, the article introduces the selected studies and the concepts derived from these studies before it presents the resulting conceptual framework. Then it discusses the validity of the conceptual framework and its implications for research. The article ends with a summary of the main conclusions of the article and introduces future work.

RELATED WORK

Since the publication of the Agile Manifesto in 2001 (Beck et al., 2001), many researchers and practitioners have worked on how to include security into agile software development. These discussions started quite early; Systematic Literature Reviews (SLRs) identify papers from the very beginning discussing security in relation to agile development (Saldanha & Zorzo, 2019). One example of such a paper is Beznosov's suggestions from 2003 on how to integrate security and eXtreme Programming (XP) (Beznosov, 2003). Although security in agile development has been a topic of discussion and research since then, and a substantial amount of literature is available on the subject (Bishop & Rowland, 2019), many authors point to the need for more research to better understand and solve the challenges today's software development projects are facing when it comes to security (Bishop & Rowland, 2019; Saldanha & Zorzo, 2019; Villamizar, Kalinowski, Viana & Fernández, 2018).

Lately, several literature reviews have been performed regarding agile software development and security, and even specifically on security requirements in agile development. In this section, we give an overview of these literature reviews and position this article in relation to these reviews. Additionally, this section describes conceptual frameworks that have already been developed related to security requirements in agile development.

Overview of Systematic Literature Reviews

In 2015, Oueslati et al. performed a literature review aimed at identifying "challenges of developing secure software using the agile approach" (Oueslati et al., 2015). Fourteen challenges were identified, and the challenges were categorized into five categories: "Software development life-cycle challenges" concerned security activities not being included in agile methods, and difficulty integrating security in every iteration due to short iteration times. "Incremental development challenges" were related to dealing with frequent changes. "Security assurance challenges" were related to documenting and testing the system in a manner expected for security assurance. "Awareness and collaboration challenges" included neglect of security requirements, lack of experience and security awareness, and challenges separating the developer and reviewer roles. "Security management challenges" concerned how added costs and a lack of incentives resulted in security not being prioritized.

In 2016, Khaim et al. studied what approaches are suggested for software security in agile, the role of the security expert in these approaches, and what kind of challenges emerge when integrating security and agile development (Khaim et al., 2016). They found that half of the studies consider integration of security into agile in a general way, 15% consider integration into Scrum, 23% XP and

12% FDD. In over half (54%) of the papers they studied, the security expert role was not specified, and the impression of Khaim et al. is that those papers that include the security expert role do not do this in a clear way and in a way that ensures involvement throughout development. Khaim et al. identified a long list of challenges, including separating the software developer and security expert roles, security assurance in the case of continuously changing code, documentation needs, refactoring, lack of security experience of developers, tracking security requirements in case of frequent changes, neglecting security requirements, unaware customers, etc.

In 2017, Alsaquaf et al. performed an SLR on quality requirements in large scale agile development in order to identify practices and proposed approaches to cope with quality requirements challenges in large scale distributed agile development (Alsaquaf, Daneva & Wieringa, 2007). Security was considered a type of quality requirement. They found that, despite many available approaches, none of the approaches they identified had been “tried out in real life settings” (Alsaquaf et al., 2017). Challenges were identified related to the techniques available (no widely accepted techniques; inadequacy of the existing techniques; traceability), the priorities made (functionality is prioritized; ignore some types of requirements; validated late; insufficient analysis) and related to the Product Owner (lack of knowledge; workload; availability; dependence). They pointed out that the challenges in large part relate to agile-specific practices and that “some characteristics of agile [requirements engineering] pitched as strengths in agile textbooks (e.g. the role of the product owner, the use of user stories) can be considered in fact as inhibitors to engineering of [quality requirements]” (Terpstra et al., 2017).

In 2019, Bishop and Rowland analyzed literature in order to understand “the effect of security practices on software development agility” (Bishop & Rowland, 2019). Additionally, they provided a taxonomy that can be used to organize and summarize work on software security in agile. They divided papers into two main categories: phase focused and phase independent. The requirements phase was identified as the phase that had received the most research attention. Still, they pointed to a need for more research, and especially empirical research, to extend the current body of knowledge related to software security in agile development.

Additionally, 2018 and 2019 saw the publication of three literature studies that specifically considered security requirements in agile development. Both Saldanha & Zorzo (2019) and Villamizar et al. (2018) performed systematic mapping studies to understand the approaches taken to handle security requirements in agile development projects, and to assess the coverage of current research on this topic. Villamizar et al. found that most approaches are related to Scrum, and that most approaches address specification and elicitation of security requirements. Both studies found that solutions typically involve modifying the agile method or introducing new artefacts or guidelines to handle security. Saldanha & Zorzo however also point to the importance of security training and its possibility to impact the security level of the software. Both systematic mapping studies identify a lack of empirical research, including empirical evaluations of existing approaches to security requirements engineering in agile development. Other research gaps identified include tool support and verification and validation of security requirements (Villamizar et al., 2018). Muneer et al. performed a systematic literature review to compare “modern requirements management techniques with classic techniques for managing Non-Functional requirements (NFRs) in agile Software Methods”, focusing primarily on security requirements as a type of NFR (Muneer, Nadeem & Kasi, 2019). Their review concludes that modern techniques have the potential to overcome some of the method weaknesses identified.

Objective of This Work

The work presented in this article is not an SLR, but a lighter and less comprehensive form of literature review with a goal to complement existing SLRs in this area. Previous SLRs have covered challenges related to security and agile (Oueslati et al., 2015), have identified the approaches covered in research literature (Khaim et al., 2016; Saldanha & Zorzo, 2019; Villamizar et al., 2018; Bishop & Rowland, 2019; Alsaquaf et al., 2017), the weaknesses and strengths of available approaches (Muneer et al., 2019), the effect on agility (Bishop & Rowland, 2019), and the role of the security

expert (Khaim et al., 2016). Most SLRs point to the need for more empirical research (Alsaquaf et al., 2017; Bishop & Rowland, 2019; Saldanha & Zorzo, 2019; Villamizar et al., 2018). We are however not aware of any SLR that gives an overview of what we can learn from the empirical research that has already taken place on security requirements work in agile development projects. This article is an attempt to fill this gap.

As a mean to give an overview of what we can learn from existing empirical research on this topic, this article combines previous findings into a conceptual framework. A conceptual framework can be defined as “a network, or ‘a plane’, of interlinked concepts that together provide a comprehensive understanding of a phenomenon or phenomena” (Jabareen, 2009). It is a product of theorization (Jabareen, 2009) and has particular benefits for designing studies as it “forces you to be selective to decide which variables are most important, which relationships are likely to be most meaningful, and, as a consequence, what information should be collected and analyzed at least at the outset” (Miles & Huberman, 1994). Thus, a conceptual framework based on existing empirical research can be used to guide future empirical research on security requirements in agile development, a type of study that is most needed. Conceptual frameworks can be “rudimentary or elaborate, theory-driven or commonsensical, descriptive or causal” (Miles & Huberman, 1994). However, in this article we take the advice from Jabareen (2009) to take an interpretative approach, rather than a causal/analytical approach; stating “[c]onceptual frameworks aim to help us understand phenomena rather than to predict them” (Jabareen, 2009).

Existing Conceptual Frameworks Related to Security Requirements in Agile Development

We are aware of two existing attempts to create conceptual frameworks related to security requirements in agile development. These have a different foundation than the conceptual framework presented in this paper.

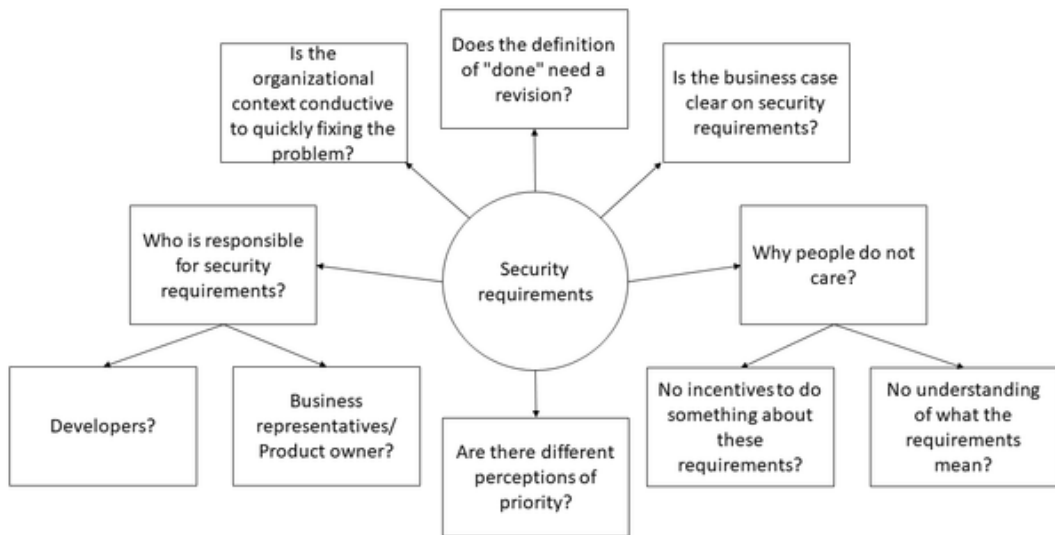
Conceptual Framework on Security Requirements as Viewed by Agile Practitioners

Terpstra et al. (2017) conducted a survey of practitioners’ postings on social media (LinkedIn) to discover how agile practitioners reason about security requirements and how they cope with this type of requirements. The analysis resulted in the identification of 21 concepts that indicate problems regarding security requirements in agile, and 15 coping strategies. The problems identified are varied, with central themes being the business value of security, the priorities that have to be made, the tendency that security gets lost in the process, and the lack of awareness and knowledge. The analysis additionally resulted in the development of a descriptive conceptual framework from the viewpoint of the development team. This conceptual framework has been redrawn in Figure 1. The boxes in this figure represent conceptual categories defined by Terpstra et al. that map to the figure in the following way:

- **Ownership of security requirements:** Who is responsible for security requirements? Developers? Business representatives/Product owner?
- **Definition of “done” (DoD):** Does the DoD need a revision?
- **Business case:** Is the business case clear on security requirements?
- **Attitude towards security requirements:** Why people do not care? No incentives to do something about these requirements? No understanding of what the requirements mean?
- **Organizational setup:** Is the organizational context conducive to quickly finding the problem?
- **Perceptions of priority:** Are there different perceptions of priority?

The conceptual framework developed by Terpstra et al. describe important influences and challenges with security requirements work in agile. It is however only based on one study.

Figure 1. Conceptual framework developed by Terpstra et al. (2017)



Conceptual Framework Showing Important Categories When Integrating Security Requirements Into Agile Development

Daneva & Wang (2018) performed a document analysis of seven “well documented agile secure development frameworks put forward by companies or non-profit industry organizations supported by companies.” Based on the practices that were part of these documents, they created the conceptual framework depicted in Figure 2. The central overarching category of the framework is “Absorb security requirements”, representing that “the development team absorbs the needs and the responsibility for

Figure 2. Conceptual framework developed by Daneva and Wang (2018)



engineering security requirements” (Daneva & Wang, 2018). What this means is represented by the other concepts in the framework:

- **Activities:** Introducing security activities (organizational and/or technical);
- **Artefacts:** Examples include abuser stories and risk assessments;
- **Roles:** Organizational or technical roles that could mirror agile specific roles (such as Scrum master or product owner), or adding security expertise to the team;
- **Competencies:** Having necessary competence, e.g. on security testing and secure architecture.

This conceptual framework presents important categories for security requirements work, as documented by key players. It however only describes current practice to the extent that the documented frameworks are used as written in these documents.

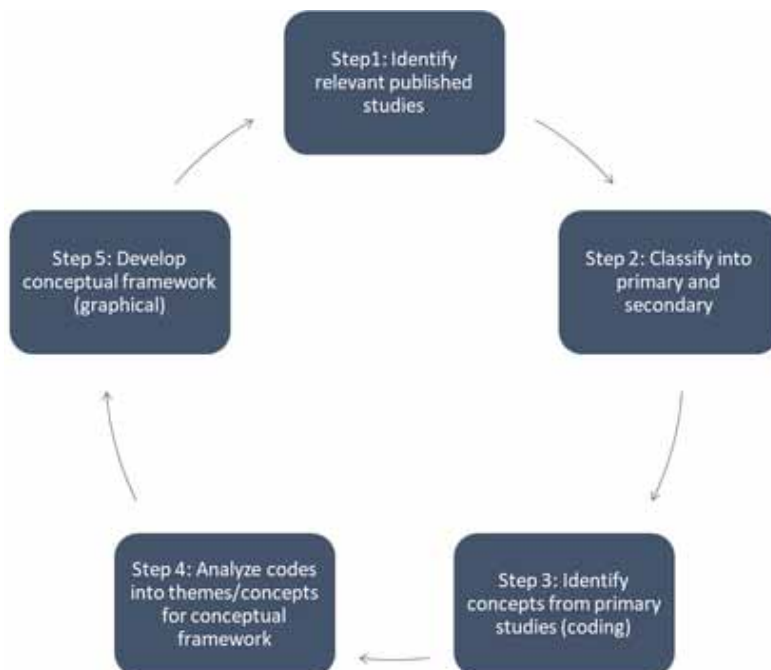
RESEARCH METHOD

In order to construct a conceptual framework based on existing empirical studies, we performed a series of phases: identifying published studies to use as a basis for the framework, analyzing the published results of these studies to identify central concepts related to security requirements work, and synthesize these concepts into a conceptual framework. In the following we describe each of these phases in more detail. An overview of our approach is given Figure 3. As can be seen from this figure, our approach was cyclical. In all we did three iterations of this cycle.

Identify Studies (Step 1)

Our goal in this step was to identify empirical studies covering aspects of security requirements work in agile development. We were primarily interested in studies performed in industry settings. We did

Figure 3. Method for constructing the conceptual framework



not aim for an SLR. Instead we used the following strategy to identify relevant studies: 1) identify relevant and recent SLRs and use them to find relevant papers, and; 2) supplement references from SLRs with searches on Google Scholar for more recent studies. We did this in three rounds and over a period of three years. The first round used the SLRs of Khaim et al. (2016) and Oueslati et al. (2015), in addition to searches on Google Scholar to identify more recent studies that were not covered by the published SLRs. We deliberately searched for empirical studies covering software security in agile, and not specifically security requirements work, because we use this term in a broader sense than eliciting and documenting security requirements. The second round used the SLR of Alsaquaf et al. (2017) as well as additional searches on Google Scholar. The third round used the SLRs of Bishop and Rowland (2019), Saldanha and Zorzo (2019), Villamizar et al. (2019) and Muneer et al. (2019).

We selected this lightweight approach to identifying empirical studies for two reasons: 1) we did not believe it necessary to repeat the identification of relevant publications already performed by other researchers, and 2) we did not have the goal to collect all relevant empirical evidence, but rather sufficient evidence to create a first version of a conceptual framework that could later be extended as new evidence is published.

The searching for and screening of literature was done by one researcher, and was done based on title and abstract. For the studies that seemed to fit the inclusion criteria (empirical studies of security requirements work in agile development), we further inspected the methods and results sections to determine if they were relevant for this study. In the end, we ended up with the papers listed in Table 1.

Analyze Study Results (Step 2 and Step 3)

Our approach was inspired by Jabareen (2009), who proposed a method called conceptual framework analysis particularly suited for multidisciplinary phenomena. Jabareen's method is based on the grounded theory model, and consists of seven phases: 1) "Mapping the selected data sources", 2) "Extensive reading and categorizing of the selected data", 3) "Identifying and naming concepts", 4) "Deconstructing and categorizing the concepts", 5) "Integrating concepts", 6) "Synthesis, resynthesis, and making it all make sense" and 7) "Validating the conceptual framework" (Jabareen, 2009). We did not follow this method in detail, as we are not aiming for a multidisciplinary framework with as extensive sources as is recommended by Jabareen. Instead we used aspects of this method adjusted to our needs. In particular, we took the advice to read and reread sources, and to categorize them based on importance (Jabareen, 2009) and relevance (Maxwell, 2013). All the included studies (Table 1) were divided into two categories: primary studies and secondary studies. The first category consists of studies where security requirements work is a main part of the topic studied, the study is done in an industry setting and with adequate research method (including adequate method description). Secondary studies are studies with results that could be relevant for security requirements work, but where security requirements is only a minor part of the overall study or where the study is not adequately described, has obvious methodological limitations or is performed in a student setting. Only the primary studies were used to identify concepts, while results from the secondary studies were used where relevant to better understand and to validate/contradict the findings in the primary studies. Concepts were identified by reading the results and discussion part of the primary studies in detail, identifying any results related to security requirements work, and adding codes to these results.

Synthesize (Step 4 and Step 5)

The method used for constructing a conceptual framework builds on the practical advice of Miles and Huberman (1994). To cite Miles and Huberman, "[s]etting out bins, naming them, and getting clearer about their interrelationships lead you to a conceptual framework." (Miles & Huberman, 1994). According to Miles and Huberman, conceptual frameworks are best done graphically, one should expect to do several iterations, and they suggest using prior theorizing and empirical research as important inputs. Additionally, they recommend that one should avoid the non-risk framework with only global level variables and "two-directional arrows everywhere" (Miles & Huberman, 1994).

Table 1. Overview of included papers and which SLRs that also include them

Identified Paper	Oueslati et al. (2015)	Khaim et al. (2016)	Alsaquaf et al. (2017)	Bishop & Rowland (2019)	Saldanha & Zorzo (2019)	Villamizar et al. (2019)	Muneer et al. (2019)
Adelyar & Norta (2016)				x			
Aydal et al. (2016)		x					
Baca & Carlsson (2011)				x	x		
Baca et al. (2015)		x			x		
Bartsch (, 2011)	x			x	x		x
Bellomo & Woody (2012)	x						
Fitzgerald et al. (2013)					x		
Ghani et al. (2014)				x	x	x	
Kongsli (2006)	x	x		x		x	
Nicolaysen et al. (2010)							
Pohl & Hof (2015)		x					
Poller et al. (2017)				x			
Rajba (2018)				x			
Renatus et al. (2015)							
Rindell et al. (2016)					x		x
Sachdeva & Chung (2017)				x		x	x
Savola et al. (2012)		x					
Terpstra et al. (2017)				x	x		x
Tøndel et al. (2017)							
van der Heijden et al., 2018				x			
Williams et al. (2009)							
Williams et al. (2010)					x		

As will be seen from the following section, we made tables of concepts from the primary sources, as suggested by Jabareen. In making these tables we integrated and synthesized concepts, as suggested by Jabareen, in an iterative manner. Using Mind Manager, initial concepts identified from the primary studies were grouped into overall concepts. This was done first for the different study types identified, and then we combined concepts from the different study types into a modified and combined table of concepts that we used to create the conceptual framework. In creating the graphical conceptual framework, we looked again to the primary studies to identify what relations between the concepts were visible in those studies.

CONCEPTS FROM EMPIRICAL STUDIES

In the following, we give an overview of the identified primary studies and the concepts identified from these studies. An overview of the primary studies can be found in Table 2, while an overview of secondary studies can be found in Table 3. We start with presenting studies that

Table 2. Overview of primary studies

Reference	Study Goal	Study Method
Adelyar & Norta (2016)	Identify challenges in the customer and developer practices	Interviews, 4 teams, team manager and 2-3 developers from each team
Baca et al. (2015)	Evaluate SEAP (more security resources in the team; incremental risk analysis)	Action research, one project, four versions, 8 development teams
Bartsch (2011)	Understand challenges and mitigations in security-critical agile development	Interviews (10 interviewees from 9 companies)
Nicolaysen et al. (2010) (interview part)	Understand whether software security was a specific concern in agile development	Six interviews with software developers
Poller et al. (2017)	Impact of external audit and training	Questionnaires, observations, document studies, interviews (15); 13 months
Terpstra et al. (2017)	Discover how agile practitioners reason about security requirements and how they cope with this type of requirements	Postings on LinkedIn, two discussion threads
Tøndel et al. (2017)	Identify risk-centric practices in software security	Interviews; 23 organizations
van der Heijden et al. (2018)	Identify security challenges in large-scale agile development	Interviews (ten interviews from five teams) in a financial organization
Williams et al. (2010)	Effect of using Protection Poker (a technique for security risk estimation)	Case study (observations, survey) in one team, 3 months

Table 3. Overview of secondary studies

Secondary Study	Reason for not Including as Primary Study
Aydal et al. (2016)	Lack of information about research method
Baca & Carlsson (2011)	Security requirements is not a main focus
Bellomo & Woody (2012)	Security requirements is not a main focus; Some information on research method is lacking
Fitzgerald et al. (2013)	Limited focus on security requirements work
Ghani et al. (2014)	Lack of information about research method
Kongsli (2006)	Experience report
Nicolaysen et al. (2010) (case study part)	Case study in research project and with unclear research method
Pohl & Hof (2015)	Evaluation with students; weak research method
Rajba (2018)	Lack of information about research method
Renatus et al. (2015)	Security requirements is not a main focus; unclear research method
Rindell et al. (2016)	Security requirements is not a main focus
Savola et al. (2012)	Limited focus on security requirements work
Sachdeva & Chung (2017)	Lack of information about research method
Williams et al. (2009)	Evaluations with students

cover software security in agile development in a broader sense, and then move on to studies of specific techniques relevant for software security work. The study by Terpstra et al. (2017), although it could be considered part of the first study category, is described in a separate section. This is because Terpstra et al. provide a conceptual framework based on their findings, thus concepts from this study have already been identified.

Studies That Provide an Overview of Security Requirements Work and Challenges

Six of the primary studies concern software security in agile development in a general sense, not tied to any particular security technique and not limited to security requirements work. Five of these studies use interviews as the mean of data collection. Adelyar and Norta performed interviews with 3-4 representatives from four teams, with the goal to identify security challenges in agile practices (Adelyar & Norta, 2016). Bartsch performed ten interviews with participants from nine companies, with the goal to “expand on the theoretical findings on security-critical agile development through an exploration of the challenges and their mitigations in typical agile development projects” (Bartsch, 2011). In the interviews, they focused on the following topics: “Customer involvement”, “Developer security awareness and expertise”, “Effects of ‘agile’ on security”, “Security practices” and “Authorization”. van der Heijden et al. performed ten interviews with varying roles in five teams, to understand challenges in large-scale agile development (van der Heijden, Broasca & Serebrenic, 2018). Nicolaysen et al. (2010) performed six interviews with software developers from different companies who were using agile methodologies. The goal of the interviews was to understand whether software security was a specific concern in agile software development. Tøndel et al. (2017) performed interviews with representatives from 23 different public organizations related to their software security practices and challenges. The goal of the study was to understand how current software organizations can work with security in a risk-centric way, and it included both people in development teams and people in information security positions in the organizations. The organizations mainly used some type of agile development practices. The study by Poller et al. is a case study using a broader set of data collection methods (Poller, Kocksch, Türpe, Epp & Kinder-Kurlanda, 2017). Poller et al. followed a product group over 13 months, starting shortly after an external security audit, and they aimed to explore how the development group’s work routines were affected by this external security audit and training.

Table 4 gives an overview of the main concepts identified from these studies. In the following we introduce the main findings from these studies in more detail.

Adelyar and Norta (2016): Challenges With Agile Practices

Adelyar and Norta identified several agile practices that posed challenges to important security principles. Frequent changes in software, different developer pairs involved, and unclear and inconsistent requirements and priorities from the customer were found to pose challenges on security, because they limited the possibility for having a system-wide view of the software, a simple design and having an ongoing development attention on security. Additionally, it made it challenging to maintain limitation of privileges and separation of privileges.

Bartsch (2011): Effects of Agile Development on Security

Bartsch found that the individuals and their relationships are highly important when it comes to security, including whether and to what extent security requirements are identified. The role of customers and developers was explored. For customers, Bartsch found that security awareness among customers was heterogenous. Half of the interviewees talked about problems that stemmed from a lack of security awareness with customers. On the other hand, one interviewee explained about a project where “the customer was very security-aware and developed very specific security requirements because the developers were rather unaware” (Bartsch, 2011). The trust relationship between the customer and the development team impacts security. Often, customers can “only state unclear security requirements leading to implicit security requirements” (Bartsch, 2011) and customers may lack the

Table 4. Overview of concepts from the studies of agile software security overall

Concept	Relevant Findings
Priorities (functionality vs. security)	<ul style="list-style-type: none"> • Functional requirements get prioritized over software security (Nicolaysen et al., 2010; Tøndel et al., 2017)
Project constraints	<ul style="list-style-type: none"> • Management commits to fixed time and budget, resulting in few resources spent on security (van der Heijden et al., 2018)
Business case for security	<ul style="list-style-type: none"> • As security was not considered a feature, it was not part of feature requests (that is, not part of expected deliveries and current priorities) (Poller et al., 2017) • Security not seen as part of working software – it costs extra time and money without providing functionality (van der Heijden et al., 2018)
Pressure (time and other tasks)	<ul style="list-style-type: none"> • Short iterations lead to pressure, which can lead to problems integrating security activities (Bartsch, 2011) • Security gets lost in daily work due to time-pressure and other tasks that need to be finished (Poller et al., 2017)
Customers and customer relations	<ul style="list-style-type: none"> • Customers' security awareness and priorities is heterogeneous and it impacts software security in the projects (Bartsch, 2011; Nicolaysen et al., 2010) • The trust relationship with customers impacts software security, as non-technical customers have a hard time comprehending security in a technical way and often trust developers to just handle this (Bartsch, 2011) • Vendors are trusted to take care of security (Tøndel et al., 2017) • Customers (Bartsch, 2011) and Product Owners (van der Heijden et al., 2018) contribute to security with their domain knowledge, even if their security awareness is low. Close involvement of the customer/Product Owner is thus recommended. • There are "unclear privileges and responsibilities between customers and developers" (Adelyar and Norta, 2016)
Individuals and their security posture and competence	<ul style="list-style-type: none"> • "The overall security in a project depends on the security expertise of the individuals, either on the customer or developer side" (Bartsch, 2011) • Architects have a potentially important role, but this is dependent on their personal initiative and interest in security. In practice, few software architects seem to have security as a main interest (Tøndel et al., 2017) • Product Owners are "often not aware enough of the added business value for performing certain security actions" (van der Heijden et al., 2018). • Developers lack intrinsic motivation for security (Poller et al., 2017) • (Lack of) software security competence impacts software security (Nicolaysen et al., 2010) • Perceived threats, in particularly related to reputation (Nicolaysen et al., 2010) and concrete threats expressed in monetary terms (Bartsch, 2011), increase security awareness. However, this increased concern for security does not necessarily lead to a commitment to software security (Nicolaysen et al., 2010). • Much of developers' security expertise is self-taught and come from news and blogs. Developers are motivated to learn security due to a feeling of responsibility for the project with a holistic development approach. (Bartsch, 2011) • There are wide variations in security awareness. Training is important (van der Heijden et al., 2018). • Organizations lack a structured approach for software security training (Tøndel et al., 2017; Baca et al., 2015; Terpstra et al., 2017) • There is a high turnover in development teams, particularly due to inclusion of external consultants. These consultants do as they are asked to, thus if they are not asked to consider security they will not pay attention to it (van der Heijden et al., 2018).
Responsibility	<ul style="list-style-type: none"> • The responsibility for identifying and deciding on security requirements for the development projects seems fragmented - no one fights for software security (Tøndel et al., 2017) • Accountability for security actions is unclear (van der Heijden et al., 2018)
Preferred security strategy	<ul style="list-style-type: none"> • Other ways to secure the system (e.g. infrastructure security) reduces the perceived need for software security (Nicolaysen et al., 2010).

continued on following page

Table 4. Continued

Concept	Relevant Findings
Legislation, audit	<ul style="list-style-type: none"> ● Legal requirements are a key driver for performing risk analysis (Tøndel et al., 2017) ● Privacy legislation can make it difficult to work according to agile principles (Nicolaysen et al., 2010) ● External audits can increase security motivation of developers (Bartsch, 2011; Poller et al., 2017)
Communication	<ul style="list-style-type: none"> ● Improved communication among developers and quality assurance impacts security motivation of developers (Bartsch, 2011) ● Intra-company competition can impact security motivation of developers (Bartsch, 2011) ● Developers may hold incorrect assumptions about managers' security priorities when these are not made explicit (Poller et al., 2017) ● Security awareness and expertise spreads between developers in informal discussions (Bartsch, 2011) ● Important decisions are made in sprint meetings, and security people are not present in these meetings (Tøndel et al., 2017) ● Security people are sometimes involved, but seem to be passive, either waiting to be invited or participating in the beginning and then leaving the project to fend for itself (Tøndel et al., 2017) ● Silo structure - security and legal competence in the organizations does not necessarily benefit development (Tøndel et al., 2017) ● There is tension between different groups, e.g. between architects and legal/security experts. Hard to make compromises. (Tøndel et al., 2017) There is a lack of understanding between information security officers and the development team; feels like "chasing different goals" (van der Heijden et al., 2018). ● Lack of documentation makes communication between the team and the security officer ineffective (van der Heijden et al., 2018) ● Security-related information should be easily available to the team (van der Heijden et al., 2018) ● Close involvement with a Security Officer is beneficial for teams, especially since this increases acceptance of security (understand why) (van der Heijden et al., 2018)
Development approach	<ul style="list-style-type: none"> ● A holistic development approach can lead to a more complete picture of the system for developers, and can impact developers' sense of responsibility for security. A more complete picture of the system can additionally, together with iterative and incremental development, lead to improved and simpler design (Bartsch, 2011). ● Team autonomy can make it more difficult for managers to prescribe security activities (Poller et al., 2017) ● Frequent changes in software requirements cause repetition of work, pressure on developers, more complex designs, illogical sequences of integration. This impacts the attention developers give to security, and make it hard to keep a system-wide view and demonstrate that the important threats have been identified and mitigated (Adelvar and Norta, 2016).
Representation of security requirements	<ul style="list-style-type: none"> ● Security requirements can be implicit or explicit. Customers can often only state implicit and unclear security requirements (Bartsch, 2011). ● Security was considered a matter of quality, and developers were expected to deal with quality matters without these being explicit and visible (Poller et al., 2017) ● Developers often derive security requirements from functional requirements. Some document them as part of Definition of Done (DoD) to make the security requirements explicit (Bartsch, 2011). ● Having a formal security requirements process can be considered too theoretical and bureaucratic (Poller et al., 2017) ● It is unclear what it means to "properly take care of security concerns", e.g. what the documentation requirements are (van der Heijden et al., 2018) ● The security requirements formulated by security management were considered too technical, but also ambiguous. From the security side there is the desire to keep the requirements generic (van der Heijden et al., 2018).
Iterative process	<ul style="list-style-type: none"> ● Security requirements are usually refined over several discussions and iterations. Functional changes as well as the complexity of security requirements can impact the need for refinement and iterations on the security requirements (Bartsch, 2011).

necessary technical expertise to understand the basis for the security measures. Thus, customers usually assume that developers take appropriate measures to ensure adequate quality. However, a majority of the interviewees mention that “irrespective of the customer’s security awareness, close customer participation improves the security requirements elicitation with their domain knowledge” (Bartsch, 2011). For developers, their individual interest in and sense of responsibility for security is important as security awareness is generally built in an informal way; security knowledge is spread as part of informal discussions and is often self-taught from news sources and blogs.

Bartsch found that the agile development approach has benefits when it comes to security, despite some well-known challenges (e.g. “neglected assurance practices from the pressure of short iterations” (Bartsch, 2011)). Agile practices can bring on a simpler software design and a more holistic development approach for the individual developer. Bartsch found that this could lead developers to feel responsible for the project, and thus increase their motivation regarding security. Compared to pre-agile development, interviewees stated that “improved communication among developers and quality assurance helped” (Bartsch, 2011) in addition to external audits and intra-company competition on quality.

In general, security requirements are refined over several iterations. Bartsch explains that in one project the authorization requirements were complicated and difficult to elicit bottom up, thus a simpler top down model was implemented that then had to be refined and adapted in production. In another project, functional changes repeatedly required security to be discussed.

Van der Heijden et al. (2018): Challenges in Large-Scale Agile Development

Of the challenges that van der Heijden et al. (2018) identified in their study, they were particularly concerned with which challenges were specific for large-scale agile development. These were “alignment of security in a distributed setting”, “developing a common understanding of roles and responsibilities”, and “integration of low-overhead security testing tools”. In addition, the study identified challenges that had been identified previously in the study performed by Bartsch (2011), and thus was considered to be challenges also in smaller-scale agile: “implementing low-overhead security documentation”, “spreading security awareness and expertise in the team”, “formulating clear security requirements”, and “fostering Product Owner commitment to security”.

Nicolaysen et al. (2010): Software Security as a Concern in Agile Development

Nicolaysen et al. found that many factors negatively impact how the need for software security is perceived and prioritized. In general functionality is given priority. About half of the customers express some security concerns, but customers’ influence on security is not necessarily positive. They give an example of this; one customer “thwarted a security solution [...] because they did not like it” (Nicolaysen et al., 2010). The studied companies have a lack of security competence, few state that they have experienced any security breaches, and in general security protection is achieved through the infrastructure (e.g. firewalls). Reputation damage is something that worries the interviewees, but the worry is not enough to commit to increased security efforts. Nicolaysen et al. state that that “[n] one of the companies had found or created any fully developed technique for integrating software security into agile software development” (Nicolaysen et al., 2010).

Tøndel et al. (2017): Risk Centric Software Security Practices

Tøndel et al. found that practices in the studied organizations were not risk based, although the organizations performed some activities that could be said to be part of a risk-based approach to security. Legal requirements were found to be an important driver for software security activities and requirements.

Responsibility for software security was fragmented in many of the studied organizations. In particular it seemed unclear where the responsibilities of security people in the organization end and where the responsibility of the development organization starts when it comes to software security.

Although the organizations might have security experts in-house, this expertise did not necessarily benefit the security work in the development projects, due to the silo structure of the organizations. People working on security or other non-functional requirements did not necessarily have a place at the table when important decisions on security were made in the projects. In many projects, involvement of security expertise was considered challenging because development was done by external vendors. The organizations offered limited formal training on software security. Software architects were pointed out as potential allies in the software security work, but with the challenge that few architects were considered to be particularly interested in security. As a result, it seemed arbitrary whether or not security was considered for the projects. In general, functionality was often prioritized over security.

Poller et al. (2017): Effects of External Security Audits on Organizational Change in Relation to Software Security

Poller et al. (2017) found that software security was considered a quality aspect among other quality aspects, and that in the studied company developers were thus expected to deal with security (as with other quality aspects) without this being made explicit and visible. This was considered by Poller et al. as a main reason for software security work not being established in the company. For developers, the feature requests represented expected deliveries, and as security was not considered a feature it was not on the feature request list. This resulted in security being perceived as not important by some developers. There was a perception that managers “would see security as being in a resource conflict with feature development” (Poller et al., 2017). The study however found that managers did not seem opposed to security, but rather that security “had not yet come to their specific attention” (Poller et al., 2017), and that it was considered a quality matter that developers were trusted to deal with as a technical issue. Additionally, security lacked visibility in the team and the developers in general lacked intrinsic motivation for security; their motivation was to “put something together and seeing it work” (Poller et al., 2017).

The study identified challenges with having autonomous and self-organizing teams in that managers had limited means of prescribing security activities. Instead they had to rely on less direct approaches, e.g. indicators or training. Developers, on the other hand, found that security got “lost in the daily work since we always have time-pressure, the release needs to be finished, tests need to be done” (Poller et al., 2017), thus they did not find time to really go deep on security. Additionally, lack of resources was considered one reason.

There was an attempt by security experts to establish a formal security requirements elicitation process, but this met resistance from managers and developers because it was considered theoretical and bureaucratic, and they were not convinced it would improve security.

Related Findings From Secondary Studies

Baca and Carlsson (2011) used interviews to evaluate the cost and benefit of the Microsoft SDL, the Digital Touchpoints and Common Criteria for agile development. They found that none of these approaches were a good match with agile development because of high cost and a lack of benefits. However, the activity of writing security requirements was endorsed, as developers believed it could help identify easy gains and help guide the project. Aydal et al. (2006) demonstrated that software security can be integrated with XP practices. In their study, security requirements were introduced rather late in the process and they found that this could lead to many changes in the system. They suggested using the Planning Game to establish security requirements within iterative and incremental development.

There is some evidence in the study of Savola, Frühwirth & Pietikäinen (2012) that indicate that regulations in a domain can impact the work on security requirements. In their study on metrics they found that “the practitioners emphasized compliance (with legal and industry regulations, customers’ needs and organizational policies), whereas 80% of researchers emphasized the metrics’ ability to

offer a high-level overview of security” (Savola et al., 2012). Rindell, Hyrynsalmi & Leppänen (2016) report on using Scrum for a regulated project with positive results. In the studied project, security was however to some extent viewed as being on the side of the project (implemented in side tracks to the main sprint cycle) and much of the extra security related work was documentation related.

Rajba (2018) presents a journey of one company in improving their security processes. Challenges identified in the beginning of this journey included complex security checklists that were considered too technical and not relevant, people being more interested in passing security reviews than making more secure applications, challenges in scoping the security work so as to not having to undertake too much at once, repetitive tasks, and a lack of documentation, security requirements and knowledge. Many of these challenges were addressed with improvements in training and security checklists, provision of templates, tool support, and improved use of an internal security review team.

Nicolaysen et al. (2010), in addition to reporting on interviews (as described above), report on a case study of a research project. Due to the domain (healthcare), security was initially given attention in the studied project. In the end, however, the resulting product had many security concerns (vulnerable to seven out of the OWASP top 10 issues), and they found that “only the functional results of the security design made it out of the backlog [...] leaving most non-functional security aspects alone in the dark” (Nicolaysen et al., 2010). Nicolaysen et al. point to some reasons for this, mainly a lack of continuity in the security experts assigned to the project, resulting in delays. Thus, the security design was not completed as planned and implementation started before security had been properly considered. Communication problems is also mentioned, although Nicolaysen et al. is not concrete on what kind of communication problems there were and how they influenced development.

Studies of Specific Techniques

Though there are several suggested techniques for identifying and working with security requirements in agile development, few of these techniques have been studied and evaluated in an industrial environment. Two of the primary studies we identified study specific techniques related to software security requirements work. Baca et al. performed an action research study at Ericsson, where they studied the effects of implementing a security-enhanced agile software development process (SEAP) (Baca, Boldt, Carlsson & Jacobsson, 2015). This process includes several software security activities (e.g. code review, penetration testing), but the study reported focused on two key aspects of SEAP: adding more security resources in the project and the development teams, and performing incremental risk analysis. The study of SEAP included one product with 88 staff members distributed among 8 development teams. Four versions of the product were considered, of which the three latest versions were developed using SEAP. Effort and identification and treatment of risk was compared between versions.

Williams et al. proposed Protection Poker (Williams, Meneely & Gegick, 2010), which is a technique for security risk estimation of requirements, as well as for security awareness building and exchange of security information in the team. The technique is particularly suited for agile teams, and Protection Poker is intended to be played in the planning meeting of every development iteration. The effects of using Protection Poker were studied in a case study including one maintenance team at Red Hat. The team had eleven participants (seven developers, three testers and one manager), and used Scrum as their development methodology. The team studied had no security expert, and the knowledge of software security varied among team members; some very knowledgeable, some relatively new to software security. The study lasted for three months with five Protection Poker sessions in total. Data collection was done using observation and a short survey.

Neither SEAP nor Protection Poker is specifically about security requirements. However, performing incremental risk analysis and doing security risk estimation could be considered part of security requirements work as defined in this paper. Table 5 gives an overview of the main concepts identified from these two studies. In addition to the findings from the studies of the techniques themselves that are used as a basis for this table, an underlying assumption is that the technique

Table 5. Overview of concepts from the studies of security requirements techniques

Concept	Relevant Findings
Incremental security analysis in the team	<ul style="list-style-type: none"> • An incremental risk analysis process improves identification and handling of risk. Security issues are solved in the team (distributed, not centralized), more detailed analysis is performed, more severe risks are identified and more risks are corrected. This leads to more cost-effective risk management. (Baca et al., 2015)
Security resources in team	<ul style="list-style-type: none"> • With security resources in the project it is possible to work distributed and solve issues in the team (Baca et al., 2015).
Security discussions in the team	<ul style="list-style-type: none"> • Using a technique for discussing security implications of functional requirements in the full team leads to improvements in software security as it results in improved spread of security knowledge and improved security skills (e.g. skills to think like an attacker), in addition to leading to identification of security needs in the project in form of security requirements, training needs and security activities (Williams et al., 2009).

itself is a factor that impacts security requirements work. In the following we explain the results of the two studies in more detail.

Baca et al. (2015): The Effect of Added Security Resources and a Distributed and Incremental Approach to Risk Assessment in an Agile Development Project

In their study, the introduction of SEAP was found to improve identification and handling of risk, and because of this, the risk management was found to be more cost-efficient than with the approach previously used by Ericsson.

Three aspects are however important to note related to this study. First, the process for risk analysis used is not explained in detail. In the study, the risk analysis of four software versions of the same product is compared, where v1.0 was developed using the traditional Ericsson approach, and v2.0 - 4.0 were developed using SEAP. With the traditional approach, risk analysis was performed once a year (per release) and involved six to eight persons for a day. With SEAP, the frequency of risk analysis was 30-40 per year, involving three to four persons for an hour each time. The scope of risk analysis with SEAP is much smaller than with the traditional approach. Another main difference between SEAP and the traditional approach is the security resources involved in the analysis and in the project in general. Traditionally the risk analysis was led by the security manager, and this role was not directly involved in the development. With SEAP, more security resources are added to the project (the equivalent of four full time positions), and one of these roles (the security master) is assigned to two or three teams (25% per team). Available time apart from security work is spent as a regular developer. The security master leads the risk analysis work. Based on the description of the risk assessment process in SEAP we thus know that the frequency is increased, the scope for each analysis is reduced, and the approach is more distributed.

Second, the reason for the identified improvement is not discussed in detail in the paper. The authors claim that a main reason for the improvement is that security issues are dealt with in a more distributed fashion, and thus more issues are solved directly by the team. Though not discussed in the paper, it should also be expected that when security resources are added to the team, the security people are more likely to understand the product and thus their analysis is likely to improve. However, there are alternative explanations that are not discussed by the authors. One example of a factor that may have influenced the results is related to the study design and its use of comparison. The traditional approach was used for v1.0, and SEAP for later versions. The ability to identify more high risks with SEAP may be due to the method, but may also be because v2.0 and up contain more risky features. This, and other alternative explanations, are not discussed by the authors.

Third, the product developed in the study was related to online money transfer and was thus considered to be security-critical. This allowed investing in the additional security resources required

by SEAP. Thus, we do not know whether or not the resources needed for SEAP can be justified for projects that are less security-critical.

Williams et al. (2010): Risk Estimation Using Protection Poker

Williams et al. found that a main effect of using Protection Poker in this case study was that software security knowledge was spread among team members, and key risks were discussed; “[d]uring Protection Poker sessions, all team members participated in the conversation - some asking questions, some sharing their software security expertise, all becoming incrementally better at thinking like an attacker with each Protection Poker session” (Williams et al., 2010). The playing of Protection Poker led to revisions of two requirements for added security, resulted in a request for education on cross-site scripting, identified the need for more security testing, etc. It was found that Protection Poker supported participation from all team members, also those with passive, quiet personalities. Note however, that we cannot say from this study whether similar results could have been achieved with another technique.

Related Findings From Secondary Studies

Protection Poker has also been studied in a trial with 50 advanced undergraduate students taking a software engineering course (Williams, Gegick & Meneely, 2009). In that study, it was found that Protection Poker resulted in more discussion and learning about software security compared to previous semesters where Protection Poker was not used. In the discussions, general lessons about security surfaced fast, e.g. discussions on input validations, common exploits, etc.

Kongsli (2006) reports on experiences with using misuse stories and automatic testing of security in the development of web applications, and points to similar benefits as Baca et al. (2015) and Williams et al. (2010) although using a different technique (misuse stories). Reported benefits include increased security awareness in the development team and team ownership of security issues. Additionally, Kongsli reports that when security is sufficiently broken down (in terms of misuse stories) it is easier to relate to and handle by developers, though with the risk of misuse story incompleteness as security concerns not directly related to a user story can be overlooked. This risk is not pointed out by Baca et al. (2015) and Williams et al. (2010). Kongsli also points out that the need for a security specialist on the team is not completely eliminated with the used security techniques.

Ghani, Azham and Jeong (2014) suggest adding the Security Backlog and the role of a Security Master to Scrum, and evaluate how this impacts agility. Results are positive, in that agility is actually found to slightly improve. This may be due to adding more security expertise and workforce.

Renatus, Teichmann and Eichler (2015) suggested a method for threat assessment in line with agile principles and a method for evaluating agility of methods, and the method itself was evaluated in one SME. Central to the method they suggested was a split between the tasks of developers without in-depth security expertise and the security curator. Security curators got the task of pre-modeling the features soon to be implemented, while the developers were tasked with figuring out how to implement the controls (Renatus et al., 2015). This is in line with SEAP and Protection Poker when it comes to an incremental approach to development but represents a slightly different approach to dealing with the need for security expertise. Renatus et al. report that it was seen as a valuable approach by the SME.

The positive effect of incremental security analysis is supported by findings from Fitzgerald, O’Sullivan & O’Brien (2013) who found that continuous compliance activities and transparency of project status facilitate risk mitigation. In particular they point to benefits of risk prioritization, as tackling the most significant risks first can improve risk mitigation. Bellomo & Woody (2012) report on an interview study among agile program managers and Accreditation Reviewers at the Department of Defense (DoD), mainly concerning high risk software where accreditation is necessary. Bellomo and Woody underline the importance to support prioritisation of security requirements and the need for security expertise being available to the team. Additionally, they advocate a risk-based incremental approach to security feature design and development, as this can mitigate the temptation to “focus on

delivering the low hanging fruits first (the easy stuff) and ignore developing the more complex, high risk capabilities” (Bellomo & Woody, 2012). Bellomo and Woody additionally found that enforced use of a security impact assessment field in the backlog increases the likelihood that security risks are continuously assessed.

Sachdeva & Chung (2017) report on a case study of two software projects, one in which security and performance requirements were handled implicitly and as an afterthought, and one in which they were identified and addressed early and added to the backlog. They found clear benefits of the latter approach. Thus, though other studies point to benefits of incremental analysis, this study point to the importance of early inclusion of security.

Pohl and Hof (2015) suggested Secure Scrum; a way to integrate security into development without changing the underlying Scrum process. Their evaluation of Secure Scrum comes with its weaknesses; relying on small student projects that lasted only a week. Bearing this in mind, the results from this study are relevant as they point to effects of having a security technique. Pohl and Hof found that security was not taken care of by the student developers that did not use Secure Scrum, but that when equipped with this technique they were able to elicit security requirements and implement some of these requirements. Security techniques can thus act as the reminder that is needed by developers to include software security.

Conceptual Framework by Terpstra et al. (2017)

Terpstra et al. (2017) created a conceptual framework based on the results they got in their study of professionals’ posts on LinkedIn. This conceptual framework is shown in Figure 1. In the following we explain the conceptual categories of Terpstra et al. in more detail.

The conceptual category *perceptions of priority* was used by Terpstra et al. to represent issues related to prioritization of security requirements at inter-iteration time. They found that customers and business representatives often push for functionality and do not prioritize security. However, developers’ priorities may be different. This is related to the conceptual category *ownership of security requirements* that was used by Terpstra et al. to represent findings that show that no role takes or is given full responsibility for security requirements in development projects. As stated by Terpstra et al. “business representatives and product owners usually have little awareness of security requirements and rarely work towards their elaboration early on” (Terpstra et al., 2017). This is supported by identified challenges such as “[t]he product owner has often too much power and instills his attitude of treating non-functional requirements”, and “[t]he product owner is sometimes acting like a business owner or stakeholder and pushes only for features” (Terpstra et al., 2017). Additionally, they found that “developers who understand risks associated with poorly treated security requirements, may not know how to communicate the possible security issues to their product owner and convincingly present him with information on how much it would cost if not fixed and if a problem arises” (Terpstra et al., 2017).

These challenges can in part be explained by findings related to the conceptual category *business case*. Some of the challenges identified by Terpstra et al. was that “[a]gile techniques are business-value driven” and “[s]ecurity is hard to ‘sell’ as a business value”. Additionally, “[s]ecurity requirements cost money to elaborate due to experts’ involvement” and “[p]eople drop security because they perceive it a fight not worth fighting” (Terpstra et al., 2017). To add to this, the conceptual category *attitude towards security requirements* was used by Terpstra et al. to represent findings that in some cases “team members ‘do not care’ about security requirements just because there is no incentive to do so (...). Or, because no one really understands completely what these requirements are” (Terpstra et al., 2017). Terpstra et al. found that using security regulation to justify the security requirements was one coping strategy used by practitioners.

The conceptual category *organizational setup* was used by Terpstra et al. to represent findings that show that the organizational culture can both help and hurt the security requirements work. Terpstra et al. in particular found that the organizations’ approaches to educating developers on software security could have an impact. Coping strategies identified include educating the business on security, raising

awareness in the development team, adding a security expert to the team, making sure the product owner is supporting security, and having cross functional streams to help not forgetting about security. Some of the challenges identified by Terpstra further explain how the agile development approach in itself may be a challenge in having security being prioritized, e.g.: “People do care about security, but do not think about it”; “Agile techniques are vulnerable for forgetting things like security” and; “Security requirements get often delivered in the last minute” (Terpstra et al., 2017). The conceptual category *definition of “done” (DoD)* was used by Terpstra et al. to represent the opinion of some of the professionals that the DoD should include security requirements. They found that security requirements often were poorly defined, and that coping strategies included integrating security into the DoD, estimates, acceptance criteria and user stories.

RESULTS: THE MAIN CONCEPTS AND THEIR RELATIONS

In the previous section we identified several concepts relevant to security requirements work as reported in the identified primary studies (see Table 3 and Table 4). Based on the concepts we identified, as well as those identified by Terpstra et al. (2017), we then identified what we consider the most important and prevalent concepts in the primary studies, and the relations between these concepts. We used this to create a conceptual framework with a graphical representation. In this section we describe the result of this work.

Main Concepts

Table 6 shows how the concepts from the primary papers have been grouped into a set of main concepts. The main concepts are as follows:

- **Teams’ security posture and competence:** The security awareness and competence of the team and the individual team members are important in remembering security, identifying the need for security, following it up with performing security activities and in having the competence needed to adequately handling the security (Terpstra et al., 2017; Bartsch, 2011; Nicolaysen et al., 2010). Benefits have been identified that can be tied to a decentralized approach to security analysis (Baca et al., 2015), but this implies commitment and capability of the development teams in doing this work;
- **Customers’ security posture and competence:** The interviews reported by Bartsch (2011) in particular, but also the interviews reported by Nicolaysen et al. (2010), show the importance the customer plays in the work on security requirements. Both these studies show that customers have the influence both to drive and hinder the work on security requirements;
- **Customer relation and involvement:** Customers have been found to provide valuable competence to the discussions on security requirements, and their competence and the trust relationship with the developers can influence how security requirements are specified (Bartsch, 2011; van der Heijden et al., 2018);
- **Business case for security:** Functionality is often prioritized over security (Nicolaysen et al., 2010; Terpstra et al., 2017). Security is in many cases not seen as part of the software or something that adds value, but rather as a cost (Terpstra et al., 2017; Poller et al., 2017; van der Heijden et al., 2018). However, legislation or audits that put requirements on security can motivate security effort (Terpstra et al., 2017; Nicolaysen et al., 2010; Bartsch, 2011);
- **Organizational culture and setup:** Several aspects with the organizational culture have been found to have an effect on security requirements work. Examples are the communication between teams and central resources on quality (Bartsch, 2011) and the organization’s approach to software security training (Terpstra et al., 2017). In addition, the organization has the potential to make decisions that impact what security resources are available in a team and the formal ownership for software security in projects;

Table 6. Main concepts

Main Concepts	Table 4 Concepts	Table 5 Concepts	Concepts Terpstra et al. (2017)
Teams' security posture and abilities	Individuals and their security posture and competence; Responsibility; Preferred security strategy	Security resources in the team	Attitude towards security
Customers' security posture and competence	Customers and customer relations; Individuals and their security posture and competence; Preferred security strategy	-	Perceptions of priority
Customer relation and involvement	Customers and customer relations	-	-
Business case for security	Priorities; Project constraints; Business case for security; Pressure; Legislation, audit	-	Business case
Organizational culture and setup	Communication; Development approach; Responsibility	Security resources in the team	Organizational setup; Ownership of security requirements
Process for making priorities on requirements	Priorities; Project constraints; Representation of security requirements	-	Perceptions of priority
Development approach	Development approach; Pressure	-	-
Security requirements elicitation approach	Iterative process,	Incremental security analysis in the team; Security discussions in the team	-
Security requirements representation	Representation of security requirements	-	Definition of "done"

- **Process for making priorities on requirements:** In agile projects, decisions on what security work to prioritize can be made without any security experts being involved and the decision can be highly dependent on the security posture of individuals (Terpstra et al., 2017);
- **Development approach:** An agile development approach can impact the security requirements positively, e.g. it has been found that developers with a holistic view of the software they develop can feel more responsibility for security (Bartsch, 2011). However, there are also known challenges (e.g. pressure of short iterations) (Bartsch, 2011; Nicolaysen et al., 2010) and frequent changes (Adelyar & Norta, 2016);
- **Security requirements elicitation approach:** Having a defined process for security to make sure security is remembered throughout the project can make a difference. Additionally, the approach selected can impact the effect of the work. As an example, approaches such as Protection Poker where the full team discusses security can lead to certain effects that would maybe not be present in a more expert oriented approach. Security requirements work is commonly considered to be iterative (Bartsch, 2011) and this should be supported by any selected elicitation approach;
- **Security requirements representation:** Security requirements often end up being implicit (Bartsch, 2011; Adelyar & Norta, 2016). Having security requirements as part of the Definition of Done is one suggested way to make them more explicit and actionable (Bartsch, 2011; Terpstra et al., 2017).

Identifying Relations Between Concepts

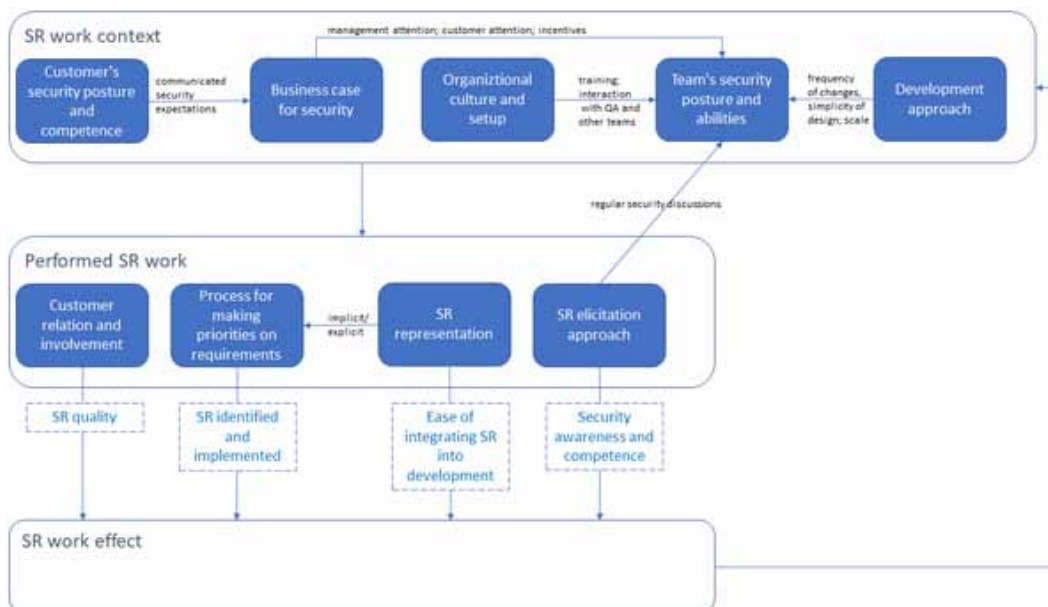
In the introduction, security requirements work was described as comprising activities to: 1) decide whether and how to identify security needs, risks or requirements for a project; 2) do the requirements elicitation; 3) communicate the identified security needs, risks or requirements, and; 4) integrate these and make priorities related to them in development. These security requirements work activities take part in a context that highly influence this work in various ways. Figure 4 depicts the conceptual framework we ended up with based on the analysis of the selected papers. Here we have divided the identified concepts into two main categories: 1) contextual factors, i.e. factors that are outside the requirements work itself, but impact the security requirements work in some way (e.g. impact the priority the security work is given, who participates, how it is done, etc.), and; 2) concepts related to the more practical aspects of the work and how it is performed (e.g. who actually participates in an activity, how the work is actually done, etc.).

In the following we describe the relations between the concepts in more detail. Additionally, we introduce a third overall category that is largely missing from the identified papers, namely that of the effect of the security requirements work.

Security Requirements Work Context

The team's security posture and abilities can be influenced by a number of factors. An obvious influence is training (Bartsch, 2011; Terpstra et al., 2017), however, this training does not need to be formal. Protection Poker is an example of a technique that has been found to spread security awareness and knowledge in a team through regular security discussions. Additionally, teams can increase their security competence through communication with quality assurance functions in the company or even a sense of competition with other teams (Bartsch, 2011). Aspects of the development work can additionally have a major impact on security posture of the team. The size and scale of the project itself can impact what type of challenges a project experience in their security work (van der Heijden et al., 2018). Adelyar & Norta (2016) found that frequent changes to software under time

Figure 4. Conceptual framework based on the selected empirical studies (SR is in the figure an abbreviation of 'security requirements')



pressure negatively impacted developers' security attention. Additionally, they found that it impacted the ability to have a simple design and made the software more complex. Bartsch (2011) found that having a holistic development approach can motivate software security and lead to simpler designs. Agile development methods can thus impact positively or negatively on the security posture and abilities of the team depending on the circumstances.

Customers' security posture and competence and the way customers are involved in the security requirements work impact software security work in many ways. It can, together with the relation between the team and customer, impact how security requirements are initially presented, especially their quality and whether they are implicit or explicit (Bartsch, 2011). Additionally, customers' security posture impacts the business case for security, e.g. by the customer making security a clear priority (Bartsch, 2011; Nicolaysen et al., 2010; Terpstra et al., 2017). The business case again influences the security posture of the individuals involved (Terpstra et al., 2017).

Performed Security Requirements Work

Several of the concepts identified fall within the category of performed security requirements work. The relations between these concepts (e.g. how the way requirements are elicited influence how they are prioritized, etc.) are however not discussed much in the papers we have studied. The main relation present is that of the impact of having implicit vs. explicit security requirements (Bartsch, 2011; Poller et al., 2017), and having security included as a feature (Poller et al., 2017).

Security Requirements Work Effect

We find that one category is largely missing from the primary and secondary studies we have identified, namely that of the effect; what makes the security requirements work useful in terms of impact. Figure 4, that shows the conceptual framework we ended up with based on the primary studies, thus includes this effect but without further concepts to help understand it. To move towards an understanding of the effect, we have however added what we understand from the sources to be potential effects of the factors included in the category 'Performed security requirements work', namely the quality of security requirements, the fact that they are identified and implemented, how easy they are to integrate into development, and the security awareness and competence that is built by doing security requirements work. Though we do not have any solid evidence to support that these are important factors characterizing the effect of security requirements work, these have support in the identified papers and can point towards factors that potentially are important for the effect of this type of work.

One effect of software security work that is somewhat available in the primary studies is that of cost. We have however not added that effect to the conceptual framework as it is not clear from the sources what the cost-benefit relationship associated with the security requirements work part is. Cost is however one likely factor of the effect of software security requirements work as well.

Figure 4 shows a possible relationship between the effect of the security requirements work and the security requirements work context. In the papers we build on, there are some pointers to the potential of security requirements work to impact the context, e.g. in form of changes in security competence and awareness among key actors, such as the Product Owner.

DISCUSSION

In the following we discuss recommendations for future research based on the conceptual framework we developed, followed by a discussion of the validity of the conceptual framework.

Implications for Research

The conceptual framework depicted in Figure 4 shows that several contextual factors influence the software security requirements work in agile development projects. This can be understood in more than one way. One possible understanding is that the contextual factors are the factors that are best

understood in the underlying research, and thus future research should aim to identify and understand also factors related to the security requirements work itself and the outcome. However, another possible understanding is that contextual factors are highly important for security requirements work and thus need to be properly understood in order to have an effective approach to software security requirements work in an agile setting. This may point to the need for more research on these factors, also keeping in mind that the contextual factors included in Figure 4 are rather complex, covering characteristics of individuals, the organization, and their interactions.

In the empirical studies we identified, the role of the approach or technique used for security requirements work is not clearly understood. Though there have been studies of different techniques and approaches, there is a difference between evaluating one technique and finding out the effects of that technique vs. understanding what makes the technique behave as it does compared to other techniques. We believe there is a need for more studies evaluating various techniques and approaches, especially in industry settings and over longer periods of time.

From the conceptual framework in Figure 4 one can see that the empirical studies we used provide limited understanding of what causes security requirements work to have effect. It may be more difficult to study and understand the effect of the work than to understand factors impacting the security requirements work, since effects may be longer term and harder to pinpoint. Still, the motivation of doing security requirements work would be an adequate level of implemented security, and if a security requirements approach does not make a significant contribution towards that then it is not worth the effort.

As can be seen from Table 7, many of the concepts we identified can be said to be directly related to the values of the Agile Manifesto (Beck et al., 2001); “[i]ndividuals and interactions over processes and tools” (V1 in Table 7), “[w]orking software over comprehensive documentation” (V2), “[c]ustomer collaboration over contract negotiation” (V3), and “[r]esponding to change over following a plan” (V4) (Beck et al., 2001). This points to the conceptual framework being related to agile development in particular, and not to other types of development approaches. However, this is not necessarily the case. Kanniah & Mahrin (2016) identified a set of factors impacting successful implementation of software development practices based on an SLR. Many of the factors they identified are related to the factors we have identified for security requirements work in agile development; the institutional context, people and action, the project context and the system development process are all represented in the conceptual framework in Figure 4. The factors identified by Kanniah and Mahrin are not considered to be specific for agile development. Based on the current evidence it is thus difficult

Table 7. Main concepts and their relation to the principles of the Agile Manifesto (Beck et al., 2001)

Identified Main Concept	V1 Individuals	V2 Software	V3 Customer	V4 Change
Teams' security posture and abilities	x			
Customers' security posture and competence	x		x	
Customer relation and involvement	x		x	
Business case for security		x		
Organizational culture and setup	x			
Process for making priorities on requirements	x			x
Development approach	x	x		
Security requirements elicitation approach	x			x
Security requirements representation	x			

to say what impact an agile development approach has on software security work, and thus what in the conceptual framework we developed are specific to agile development, even if the conceptual framework is entirely based on studies done on projects and companies using some kind of agile development approach. Of the four agile principles, it is the value “Individuals and interactions over processes and tools” (Beck et al., 2001) that seems to be most influential on security requirements work. Of the ten factors that Kaniah and Mahrin later identified as the most influential (Kanniah & Mahrin, 2018), a majority can also be considered to be concerned with individuals and interactions.

To sum up, there is a need for a deeper understanding of software security work in agile development. Especially there is a need to understand better what factors are important for the effect of the work, and to understand the role of the particular approach in bringing about this effect. However, a conceptual framework has a role not only in directing research priorities but also to “identify potential validity threats to your conclusions” (Maxwell, 2013). It is clear that contextual factors are important and influence software security work in agile development in many ways, especially factors concerning individuals and their interactions. Thus, understanding these and taking these factors into account in future studies is essential in order to properly understand the findings. Thus, the conceptual framework can be used to guide future research priorities, but also be input to planning and analysis of future empirical studies.

Validity of the Conceptual Framework

The conceptual framework presented in this paper is based on nine empirical studies; five interview studies addressing software security in agile (Adeyar and Norta, 2016; Bartsch, 2011; Nicolaysen et al., 2010; Tøndel et al., 2017; van der Heijden, 2018), one case study on the impact of external security audits on development (Poller et al., 2017), two evaluations of approaches or techniques relevant for security requirements work (Baca et al., 2015; Williams et al., 2010) and one analysis of professionals’ postings on LinkedIn related to security requirements in agile (Terpstra et al., 2017). These nine studies together address the topic of security requirements work in agile from varying perspectives and using varying methods, something that can be considered a strength. Still, the nine primary studies we build on can be considered to be rather few, thus we have used a set of secondary studies to improve understanding of the concepts identified from the primary studies.

As previously explained, we decided not to do a comprehensive and systematic search for literature, as one would expect if doing an SLR. We made the initial assessment that given the recent SLRs on software security in agile (Khaim et al., 2016; Oueslati et al., 2015) that we could use as a basis for this work, it was not worthwhile to do a comprehensive search for literature. At later stages in the process we used even more recent SLRs (Alsaquaf et al., 2017; Bishop & Rowland, 2019; Saldanha & Zorzo, 2018; Villamizar et al., 2018) to add to the initial selection of papers. Deciding not to do a full SLR is a weakness of our approach, and it can potentially have impacted the conceptual framework we ended up with, as more identified studies could have resulted in more and/or different concepts and relations between them. However, we never intended this conceptual framework to be a complete and “finalised” conceptual framework, but rather a work in progress that should be improved as more research becomes available (Maxwell, 2013). We would additionally like to point out that the SLRs we used to identify papers seem to vary in what papers are included (see Table 1), something that may indicate challenges in identifying all relevant papers also when doing an SLR. Several of the SLRs we have used as a basis state that the current number of published empirical studies on software security in agile development is rather low (Alsaquaf et al., 2017; Bishop & Rowland, 2019; Saldanha & Zorzo, 2018; Villamizar et al., 2018), thus identification of a high number of studies should not be expected regardless of method for identifying studies.

The conceptual framework presented in this paper is based solely on published empirical studies. Restricting ourselves to only using such studies as a basis for the conceptual framework represents a narrowing of focus, ignoring other sources of knowledge of security requirements work such as unpublished results and the general experiences of researchers and practitioners (Maxwell, 2013;

Robson, 2011). Also, for this reason, this conceptual framework is to be considered work in progress, and something that will need to be refined including more sources.

Relying on published empirical studies additionally pose limitations in that we only have access to as much information about the studies as is available in the published papers. For the study category consisting of more general studies, we would generally have benefited from more information on study context as this would help us understand the results and the selected concepts in more depth. For the studies of specific techniques, the results and thus the concepts are highly related to the specifics of the techniques; if other techniques had been studied it is likely that other concepts would have emerged from the results. It is hard to know what is the effect of the studied approaches (SEAP, Protection Poker) compared to that of other techniques, i.e., which effects are due to the particular way of working in the technique, and which are due to other factors.

Although the work of creating this conceptual framework has been done in a structured way, there is always an element of creativity also in scientific work (Collins, 2019). In this work, the coding of results from the primary sources (step 3), the reorganizing of these codes into themes/concepts for the conceptual framework (step 4) and the development of the graphical representation of the conceptual framework (step 5) all represent some form of creative work, although based on a structured process and although we have aimed to preserve the link between the resulting framework and the findings in the primary sources. It is likely that other researchers would make slightly different categorizations and end up with a different graphical representation of the conceptual framework. In many cases the concepts we ended up with using, both the initial concepts identified based on the primary studies (Table 4 and Table 5) and the main concepts used in the conceptual framework (Table 6), are somewhat overlapping. To illustrate, the security discussions in the teams that are part of Protection Poker in many ways represent one form of incremental security analysis done by the team, and such a discussion influences the security resources in the team. The concepts we ended up with using represent our best effort to group key findings from the primary studies into meaningful and related concepts. The concepts and the framework we ended up with should however not be viewed as a final version, but as a starting point and something that can be improved upon as more empirical research becomes available.

Further evaluation of this conceptual framework is needed. In its current state, the conceptual framework has been primarily developed by one researcher. Future work includes discussing the conceptual framework with more colleagues and validating and improving the conceptual framework when new evidence becomes available. Note however that the concepts in the framework bear similarities to the categories of challenges identified by Oueslati et al. (2015), especially to their categories “[a]wareness and collaboration challenges” and “[s]ecurity management challenges” and to challenges identified by Khaim et al. (2016) and Alsaquaf et al. (2017). Thus, the factors we have identified have been pointed out also by other researchers aiming to understand challenges relating to software security in agile development or quality requirements in agile development. Compared to the conceptual framework developed by Daneva and Wang (2018), it integrates their key concepts of activities, competencies, roles and artefacts, although in a slightly different way. Also, note that the last iteration of the framework that included adding two more primary studies (Adelyar and Norta, 2016; van der Heijden et al., 2018) resulted in only minor updates to the final concepts and to the conceptual framework. Thus, we have reason to believe that this conceptual framework is able to cover the key findings in current empirical research on software security requirements work in agile development.

In the Research Method section, we restated the recommendation from Miles and Huberman (1994) to avoid a non-risk framework with only global level variables and two-directional arrows. The conceptual framework we have presented in this paper is not a non-risk framework, but could be said to be a low-risk framework with many high level concepts and mainly high-level relations between the concepts. This is in many ways a result of limited studies to use as a basis for the conceptual framework. Both Maxwell (2013) and Robson (2011) recommend an inclusive approach at the initial stage. However, the conceptual framework should become more focused as it is refined (Maxwell,

2013). Thus, future revisions should move towards more specific concepts, and even excluding concepts that are less important. Revisions can be made based on new data becoming available, or could utilize other sources like experience, a broader set of literature, and thought experiments (Maxwell, 2013; Robson, 2011).

CONCLUSION AND FUTURE WORK

This paper suggests a conceptual framework for software security requirements work in agile development, with the motivation to increase understanding of this type of work and guide further research. The conceptual framework is based on published empirical studies covering aspects of software security requirements work in agile in an industrial setting. The results point to a need for further empirical studies in this area, especially to improve understanding of factors important for gaining impact from the work on software security requirements in agile projects, as this is largely missing in current work. There is additionally a need for understanding to what extent the concrete approach adopted for security requirements work shape the impact of this work given varying contexts. This would help practitioners in deciding what methods to adopt for their particular case. Contextual factors seem to be highly influential on the way security requirements are treated in current software projects. Thus, these are important to take properly into account in future empirical research studies, especially in plans for data collection and in the analysis phase.

In our own work, we are in the process of using this conceptual framework as an input to planning and analysis of ongoing case studies related to software security requirements work in agile software development. Especially, we plan to use the conceptual framework to provide some structure to the analysis. Additionally, we plan to use the results of the ongoing and future case studies to improve this conceptual framework.

ACKNOWLEDGMENT

This work was supported by the SoS-Agile: Science of Security in Agile Software Development project, funded by the Research Council of Norway (grant number 247678).

REFERENCES

- Adelyar, S. H., & Norta, A. (2016, September). Towards a secure agile software development process. In *Proceedings of the 2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)* (pp. 101-106). IEEE. doi:10.1109/QUATIC.2016.028
- Alsaqaf, W., Daneva, M., & Wieringa, R. (2017). Quality requirements in large-scale distributed agile projects—a systematic literature review. In *Proceedings of the International working conference on requirements engineering: Foundation for software quality* (pp. 219–234). Academic Press. doi:10.1007/978-3-319-54045-0_17
- Aydal, E. G., Paige, R. F., Chivers, H., & Brooke, P. J. (2006). Security planning and refactoring in extreme programming. In *Proceedings of the International conference on extreme programming and agile processes in software engineering* (pp. 154–163). Academic Press. doi:10.1007/11774129_16
- Baca, D., Boldt, M., Carlsson, B., & Jacobsson, A. (2015). A novel security-enhanced agile software development process applied in an industrial setting. In *Proceedings of the 10th international conference on availability, reliability and security (ARES)* (pp. 11–19). Academic Press. doi:10.1109/ARES.2015.45
- Baca, D., & Carlsson, B. (2011). Agile development with security engineering activities. In *Proceedings of the 2011 international conference on software and systems process* (pp. 149–158). Academic Press.
- Bartsch, S. (2011, Aug). Practitioners' perspectives on security in agile development. In *Proceedings of the 2011 sixth international conference on Availability, reliability and security (ARES)* (p. 479-484). doi:10.1109/ARES.2011.82
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . others (2001). Manifesto for agile software development. Retrieved from <http://www.agilemanifesto.org>
- Bellomo, S., & Woody, C. (2012). DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers. Carnegie-Melon University.
- Beznosov, K. (2003, October). Extreme security engineering: On employing XP practices to achieve 'good enough security' without defining it. In *Proceedings of the First ACM Workshop on Business Driven Security Engineering (BizSec)*. Academic Press.
- Bishop, D., & Rowland, P. (2019). Agile and secure software development: An unfinished story. *Issues in Information Systems*, 20(1).
- Collins, H. (2019). *Forms of Life: The Method and Meaning of Sociology*. MIT Press.
- Daneva, M., & Wang, C. (2018, August). Security requirements engineering in the agile era: How does it work in practice? In *Proceedings of the 2018 IEEE 1st International Workshop on Quality Requirements in Agile Projects (QuaRAP)* (pp. 10-13). IEEE.
- Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119, 87–108. doi:10.1016/j.jss.2016.06.013
- Fitzgerald, B., Stol, K.-J., O'Sullivan, R., & O'Brien, D. (2013). Scaling agile methods to regulated environments: An industry case study. In *Proceedings of the 2013 international conference on software engineering* (pp. 863–872). Academic Press. doi:10.1109/ICSE.2013.6606635
- Ghani, I., Azham, Z., & Jeong, S. R. (2014). Integrating Software Security into Agile-Scrum Method. *TIIS*, 8(2), 646–663. doi:10.3837/tiis.2014.02.019
- Hanssen, G. K., Stålhane, T., & Myklebust, T. (2018). *SafescrumOR -agile development of safety-critical software*. Springer.
- Heeager, L. T., & Nielsen, P. A. (2018). A conceptual model of agile software development in a safety-critical context: A systematic literature review. *Information and Software Technology*, 103, 22–39. doi:10.1016/j.infsof.2018.06.004
- Howard, M., & Lipner, S. (2006). *The security development lifecycle*. Microsoft Press.

- Jabareen, Y. (2009). Building a conceptual framework: Philosophy, definitions, and procedure. *International Journal of Qualitative Methods*, 8(4), 49–62. doi:10.1177/160940690900800406
- Kanniah, S. L., & Mahrin, M. N. (2016). A review on factors influencing implementation of secure software development practices. *International Journal of Computer and Systems Engineering*, 10(8), 3032–3039.
- Kanniah, S. L., & Mahrin, M. N. (2018). Secure software development practice adoption model: A delphi study. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(2-8), 71–75.
- Khaim, R., Naz, S., Abbas, F., Iqbal, N., & Hamayun, M. (2016). A review of security integration technique in agile software development. *International Journal of Software Engineering and Its Applications*, 7(3).
- Kongsli, V. (2006). Towards agile security in web applications. In *Companion to the 21st ACM SIGPLAN symposium on object-oriented programming systems, languages, and applications* (pp. 805–808). ACM.
- Leffingwell, D. (2010). *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.
- Maxwell, J. A. (2013). *Qualitative research design: An interactive approach* (Vol. 41). Sage publications.
- McGraw, G. (2004, March). Software security. *Security & Privacy*, 2(2), 80–83. doi:10.1109/MSECP.2004.1281254
- McGraw, G. (2006). *Software Security: Building Security In*. Addison-Wesley.
- McGraw, G., Migués, S., & West, J. (2018). *BSIMM 9*. Synopsys, Inc.
- Microsoft. (2009, June 30). Security development lifecycle for agile development, version 1.0.
- Microsoft. (n.d.). Microsoft security development lifecycle (No. Accessed 2019.08.07). Retrieved from <https://www.microsoft.com/en-us/SDL>
- Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Sage.
- Muneer, S. U., Nadeem, M., & Kasi, B. (2019). Comparison of modern techniques for analyzing NFRs in Agile: A systematic literature review. *Journal of Software Engineering Practice*, 3(3), 1–12.
- Nicolaysen, T., Sassoon, R., Line, M. B., & Jaatun, M. G. (2010). Agile software development: The straight and narrow path to secure software? *International Journal of Secure Software Engineering*, 1(3), 71–85. doi:10.4018/jsse.2010070105
- Oueslati, H., Rahman, M. M., & ben Othmane, L. (2015). Literature review of the challenges of developing secure software using the agile approach. In *Proceedings of the 10th international conference on availability, reliability and security (ARES)* (pp. 540–547).
- OWASP. (n.d.). Software assurance maturity model - a guide to building security into software development. version 1.5 (Tech. Rep.). *Open Web Application Security Project*.
- Peeters, J. (2005). Agile security requirements engineering. In *Proceedings of the Symposium on requirements engineering for information security*. Academic Press.
- Pohl, C., & Hof, H.-J. (2015). Secure scrum: Development of secure software with scrum.
- Poller, A., Kocksch, L., Türpe, S., Epp, F. A., & Kinder-Kurlanda, K. (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* (pp. 2489–2503). Academic Press. doi:10.1145/2998181.2998191
- Rajba, P. (2018, August). Challenges and mitigation approaches for getting secured applications in an enterprise company. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (pp. 1-6). Academic Press. doi:10.1145/3230833.3233276
- Renatus, S., Teichmann, C., & Eichler, J. (2015). Method selection and tailoring for agile threat assessment and mitigation. In *Proceedings of the 10th international conference on availability, reliability and security (ARES)* (pp. 548–555). Academic Press. doi:10.1109/ARES.2015.96

Rindell, K., Hyrnsalmi, S., & Leppänen, V. (2016, August). Case study of security development in an agile environment: building identity management for a government agency. In *Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES)* (pp. 556-563). IEEE. doi:10.1109/ARES.2016.45

Robson, C. (2011). *Real World Research* (3rd ed.). John Wiley & Sons.

Sachdeva, V., & Chung, L. (2017, January). Handling non-functional requirements for big data and IOT projects in Scrum. In *Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence* (pp. 216-221). IEEE. doi:10.1109/CONFLUENCE.2017.7943152

Saldanha, L. R., & Zorzo, A. (2019). Security requirements in agile software development: a systematic mapping study. Pontifical Catholic University of Rio Grande Do Sul, 2019, 32p.

Savola, R. M., Frühwirth, C., & Pietikäinen, A. (2012). Risk-driven security metrics in agile software development-an industrial pilot study. *J. UCS*, 18(12), 1679–1702.

Terpstra, E., Daneva, M., & Wang, C. (2017). Agile practitioners' understanding of security requirements: Insights from a grounded theory analysis. In *Proceedings of the 2017 IEEE 25th international requirements engineering conference workshops (REW)* (pp. 439–442). IEEE Press.

Tøndel, I. A., Jaatun, M. G., Cruzes, D. S., & Moe, N. B. (2017). Risk centric activities in secure software development in public organisations. *International Journal of Secure Software Engineering*, 8(4), 1–30. doi:10.4018/IJSSE.2017100101

van der Heijden, A., Broasca, C., & Serebrenik, A. (2018, October). An empirical perspective on security challenges in large-scale agile software development. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1-4). Academic Press. doi:10.1145/3239235.3267426

Villamizar, H., Kalinowski, M., Viana, M., & Fernández, D. M. (2018, August). A systematic mapping study on security in agile requirements engineering. In *Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 454-461). IEEE. doi:10.1109/SEAA.2018.00080

Williams, L., Gegick, M., & Meneely, A. (2009). Protection poker: Structuring software security risk assessment and knowledge transfer. In *Proceedings of the International symposium on engineering secure software and systems* (pp. 122–134). Academic Press.

Williams, L., Meneely, A., & Shipley, G. (2010). Protection poker: The new software security game. *IEEE Security and Privacy*, 8(3), 14–20. doi:10.1109/MSP.2010.58

Inger Anne Tøndel is a PhD candidate at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, and a research scientist at SINTEF Digital in Trondheim, Norway. She received the M.Sc. degree in Telematics from NTNU in 2004. Her research interests include software security, security requirements, information security risk management, cyber-insurance and smart grid cyber security.

Martin Gilje Jaatun is a Senior Scientist at SINTEF Digital in Trondheim, Norway. He graduated from the Norwegian Institute of Technology (NTH) in 1992, and received the Dr. Philos degree in critical information infrastructure security from the University of Stavanger in 2015. He is an adjunct professor at the University of Stavanger, and was Editor-in-Chief of the International Journal of Secure Software Engineering (IJSSE). Previous positions include scientist at the Norwegian Defence Research Establishment (FFI), and Senior Lecturer in information security at the Bodø Graduate School of Business. His research interests include software security, security in cloud computing, and security of critical information infrastructures. He is vice chairman of the Cloud Computing Association (cloudcom.org), vice chair of the IEEE Technical Committee on Cloud Computing (TCCLD), an IEEE Cybersecurity Ambassador, and a Senior Member of the IEEE.