Literature Review of the Challenges of Developing Secure Software Using the Agile Approach

Hela Oueslati
Technical University Darmstadt
Darmstadt, Germany
hela.oueslati@stud.tu-darmstadt.de

Mohammad Masudur Rahman
Technical University Darmstadt
Darmstadt, Germany
mohammadmasudur.rahman
@stud.tu-darmstadt.de

Lotfi ben Othmane
Secure Software Engineering Group
Fraunhofer SIT
Darmstadt, Germany
lotfi.ben.othmane@sit.fraunhofer.de

***Abstract*—A set of challenges of developing secure software using the agile development approach and methods are reported in the literature. This paper reports about a systematic literature review to identify these challenges and evaluates the causes of each of these challenges, with respect to the agile values, the agile principles, and the security assurance practices. We identified in this study 20 challenges, which are reported in 10 publications. We found that 14 of these challenges are valid and 6 are neither caused by the agile values and principles, nor by the security assurance practices. We also found that 2 of the the valid challenges are related to the software development life-cycle, 4 are related to incremental development, 4 are related to security assurance, 2 are related to awareness and collaboration, and 2 are related to security management. These results justify the need for research to make developing secure software smooth.**

I. Introduction

Companies commonly use agile development methods, such as Scrum [28] and Extreme Programming (XP) [22] to develop their evolving software. These methods are associated with better developers productivity, product quality, and customers satisfaction than the waterfall methods [9]. They embrace requirement changes, prefer frequent deliveries, and their practices do not include security engineering activities. These characteristics, and others, make developing secure software using these methods challenging [25]. For instance, it is difficult to implement verification gates in the processes that implement these methods because the cost of these gates is very high, if they were to be repeated several times during the development of the project [6].

There are several published papers that discuss the challenges of developing secure software using agile development methods; that is, the problems that make developing secure software using the agile approach difficult. For instance, Benznosov and Kruchten evaluated the mismatches between security assurance methods/techniques and agile practices [7]. In response to the challenges, several approaches and methods for developing secure software using agile methods have been proposed, e.g., [29], [6]. But, there is currently no evaluation of the validity of the reported challenges.[1]

This paper aims to address the questions: what are the challenges of developing secure software using the agile methods that have been proposed in the literature? And are these challenges valid? There are currently no systematic literature reviews that address these questions. A systematic literature review is a mean to identify, analyze, and interpret the available evidence (from publications) relevant to a research question by using a sound approach [18], [31]. The answers to both questions should contribute to identifying the pending research challenges that the community shall address so organizations can use the agile methods to develop secure software.

The paper reports about a study that examines the validity of the challenges of developing secure software using the agile development methods. It summarizes the challenges reported in 10 publications and evaluates their validity with respect to a set of agile development criteria and developing secure software criteria. It is organized as follows. First we provide a short background about the agile approach and development of secure software in Section II. Then, we describe in Section III the research method that we used to identify and validate the challenges. Next, we present in Section IV the challenges that we identified from the publications that we selected and analyze the validity of the identified challenges in Section V. We discuss the limitations and impacts of the study afterwards in Section VI and conclude the paper in Section VII.

II. Background

This section provides an overview of Agile Software Development (ASD) approach and of developing secure software.

A. Overview of Agile Development Approach

Seventeen software developers met on February 2001 in the Wasatch Mountains of Utah, USA to try to find a common ground about their perception of software development [4]. They agreed on four values that the software methods they created share in a manifesto and named the approach they created Agile Software Development. Then, they developed twelve supporting principles for the manifesto. Figure 1 shows the manifesto and Table I lists the principles.

We note that there is a disagreement among researchers on whether the concept "working software" includes only the software functionalities or it includes by default also the quality requirements, such as security requirements. In this evaluation, we use "working software," without considering quality requirements, unless requested by the customers.

[1] See Section III-B for the definition of validity.

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools (V1)
Working software over comprehensive documentation (V2)
Customer collaboration over contract negotiation (V3)
Responding to change over following a plan (V4)

That is, while there is value in the items on
the right, we value the items on the left more.

Fig. 1. Manifesto for Agile software development [4].

TABLE I
PRINCIPLES FOR AGILE SOFTWARE DEVELOPMENT [4].

Code	Principle
P1.	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
P2.	Welcome changing requirements, even late in development. Agile processes harness change for the customers competitive advantage.
P3.	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
P4.	Business people and developers must work together daily throughout the project.
P5.	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
P6.	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
P7.	Working software is the primary measure of progress.
P8.	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
P9.	Continuous attention to technical excellence and good design enhances agility.
P10.	Simplicity–the art of maximizing the amount of work not done is essential.
P11	The best architectures, requirements, and designs emerge from self-organizing teams.
P12.	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The values and principles of the agile approach are implemented by several methods including: Scrum, XP, Agile Modeling (AM) [2], and Feature-Driven Development (FDD) [26]. Thus, the methods enable producing potentially shippable working software at regular intervals [2] named *iterations* (a.k.a. *cycles*), provide customers high value features (customers-valued product functionalities) in short time and accommodate several classes of software, such as Web applications [14]. It applies a greedy-like approach with incomplete information for selecting functionalities to develop.[2]

Often, the agile approach is understood to be a development philosophy. However, the methods that implement the approach use "kind of" control theory mechanisms that guide the teams to project completion [23]. This results to several advantages. First, it reduces the chance of project failure because it enables early detection of gaps between business expectations and developers understanding. Second, it enables discovery of customer needs rather than customer wishes since customers can observe demos of the product while being developed and can adapt the requirements based on their needs. Third, it enables early discovery of technical barriers since the developers experiment their ideas and use the experiment results to adapt the system architecture and work plan.

[2]The greedy approach chooses the locally optimal option with the hope to obtain the optimal global solution.

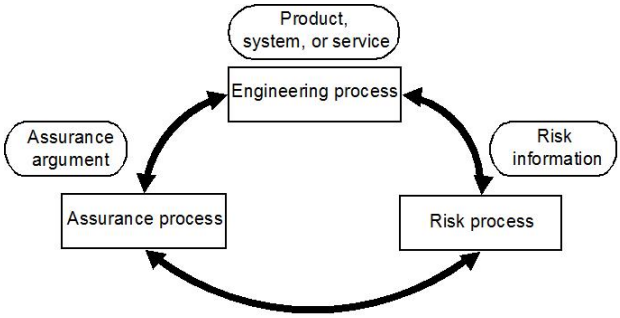

Fig. 2. CMM security engineering process.

B. Overview of developing secure software

Goertzel et al. [12] define *Secure software* as: software that cannot be intentionally subverted or forced to fail; it remains correct and predictable in spite of intentional efforts to compromise that dependability. Similarly McGraw [24] define secure software as: software that continues to function correctly under malicious (intended) attacks.

A reference model for engineering secure software is System Security Engineering-Capability Maturity Model (SSE-CMM) [15], shown in Figure 2. The process has three sub processes: risk process, engineering process, and assurance process. The risk process assesses the risk to the system. It includes practices for identifying security vulnerabilities and threats along with their impacts and occurrence likelihood. The security engineering process determines and implements solutions to the threats. It includes practices for specifying security needs and creating solutions that address the risks. The security assurance process provides confidence that the implemented security solutions reduce the risks. It includes practices for collecting evidences using testing, verification, and validation; and for argumentation that security requirements are addressed and risks are mitigated.

III. REVIEW AND ANALYSIS METHODOLOGY

We conducted a systematic literature review to identify the challenges of developing secure software using the agile approach and methods and we evaluated the validity of the identified challenges. The descriptions of both literature review and evaluation tasks follow.

TABLE II
SECURITY RISKS AND SECURITY ASSURANCE PRACTICES

Code	Assurance practice	Description
S1	Specify security policies	Identify the security needs in order to meet legal, policy, and organizational requirements for security considering the security objectives. The needs (policies) become the security claims of the software.
S2	Security training	Train the developers of the software on using cryptographic libraries and security techniques to develop security features, use security coding standards, performing security code reviews, using tools for static and dynamic analysis of source code, and performing security testing.
S3	Identify vulnerabilities	Identify and categorize the flaws of the software. These flaws are either: (1) source code security vulnerabilities, such as buffer overflows, and non-validated input or (2) weaknesses related to the software architecture, such as sending data unprotected in a public network [24].
S4	Assess security risks	Identify the security threats to the software and assess their likelihoods of occurrences and impacts.
S5	Verify and validate the security	Perform security review, static analysis, dynamic analysis of source code, security architecture review, functional testing of the security features, and penetration testing (and possibly fuzz testing). Develop arguments to show that the customer′s security needs are met. An assurance argument is a set of stated assurance objectives that are supported by assurance evidences derived from multiple sources, such as security testing and security review of code source [15].

A. Systematic literature review

We applied in this review the method proposed by Kitchenham et al. [18], [17]. The description of the activities follows.

Research question identification. The first research question investigated in this study is:

RQ1: What are the challenges of developing secure software using the agile approach?

Paper search process. We used in May 2014, the IEEE and ACM libraries to search for papers with the keywords "secure software agile." We read the titles and abstracts of the papers that the search engines found and we selected 28 papers that we found are potentially related to our research question.

Paper selection. We read the papers that we initially selected and we found only 6 papers relevant to our research question. In this selection we excluded the papers that list challenges but do not argue about them because we considered the claims as weak. We looked at the references of the papers and their citations which allowed finding 3 technical reports and 1 journal paper that are related as well to our research questions. We selected in total 10 papers for further processing.

We choose to select the papers that discuss both the challenges of developing secure software using the agile approach and the papers that discuss the agile software development methods. This implies that we may have challenges that apply to a specific agile development method but not to the agile approach. This is not a problem in our case because we already evaluate the identified challenges with respect to characteristics of the agile development approach in Section V.

Data extraction process. Two researchers read the papers and marked the text relevant to the challenges of developing secure software using the agile development approach or methods in each of these papers. In some papers the authors mention explicitly the challenges and in some other papers each researcher had to summarize the essence of the given passage using a short phrase, or a code [27]. Then, both researchers met to match their findings and discuss the differences.

We report the results of the systematic review in Section IV.

B. Evaluation of the identified challenges

In this section, we define the *validity* of the identified challenges and the method that we used to evaluate them.

Research question identification. The second research question investigated in this study is:

RQ2: Are these challenges *valid*?

A challenge for secure software development method is *valid* if either it is caused by one or many of the characteristics of the agile development approach or is caused by one or many of the characteristics of developing secure software. We describe in the following the evaluation method.

Evaluation of the challenges with respect to the agile development approach. The agile development approach is based on 4 values and 12 principles as described in Section II. A challenge is valid with respect to the agile development approach if it is caused by one or many agile values and/or principles. We use the following question to evaluate whether a challenge is valid or not: What are the ASD values or principles that cause the challenge?"

Evaluation of the challenges with respect to developing secure software. Developing secure software requires assuring that the software complies with its security policies. This requires performing correctly a set of activities that support producing secure software, as is proposed by SSE-CMM [15]. We propose the use of the five security practices of Table II as criteria for evaluating challenges for developing secure software, which we derived from the SSE-CMM process. Table II describes these criteria. These evaluation criteria do not include secure software administration because we consider that the environment is related to the deployment

TABLE III
CLASSIFICATION OF THE SECURITY CHALLENGES.

Code	Challenge	Source
Software development life-cycle challenges		
CH1.1	Security requirements elicitation activity is not included in the agile development methods	[1][8][30][11]
CH1.2	Risks assessment activity is not included in the agile development methods	[8][30][11]
CH1.3	Security related activities need to be applied for each development iteration	[7][8][11]
CH1.4	Iteration time is limited and may not fit time consuming security activities	[3][16]
Incremental development challenges		
CH2.1	Refactoring practice breaks security constraints	[7]
CH2.2	Changes of requirements and design breaks system security requirements	[30]
CH2.3	Continuous code changes makes completing the assurance activities difficult	[7][30]
CH2.4	Requirement changes makes the trace of the requirements to security objectives difficult	[3]
Security assurance challenges		
CH3.1	Security assessment favors detailed documentation	[1][7][20][32]
CH3.2	Tests are, in general, insufficient to ensure the implementation of security requirements	[7]
CH3.3	Tests do no cover in general, all vulnerability cases	[30]
CH3.4	Security tests are in general difficult to automate	[32]
CH3.5	Continuous changing of the development processes (to support lesson learned) conflicts with audit needs of uniform stable processes	[3]
Awareness and collaboration challenges		
CH4.1	Security requirements are often neglected	[3]
CH4.2	Developers lack experience on secure software	[1][3][30][32]
CH4.3	Customers lack security awareness	[1][3]
CH4.4	Developer role must be separate from security reviewer role to have objective results	[20][32]
Security management challenges		
CH5.1	Security activities increases the cost of the software	[1]
CH5.2	There are no incentive for organizations to develop security features in early increments	[3][5]
CH5.3	Organizations compromise security activities to accommodate accelerated releasing schedule	[1]

activities.[3] A challenge is valid with respect to the secure software development requirements if it is caused by one or many of the secure software assurance practices. We use the following question to evaluate whether a challenge is valid or not: What are the secure software assurance practices that cause the challenge?"

IV. CHALLENGES OF DEVELOPING SECURE SOFTWARE USING THE AGILE METHODS

This section answers the question RQ1. The systematic review resulted into identifying 20 challenges for developing secure software using the agile approach, which we classified into 5 categories. A summary of the challenges is provided in Table III. Their description follows.

Software development life-cycle challenges. ASD methods enable developing software in successive iterations [21]. These methods do not integrate security requirements elicitation activities (CH1.1) [30], [1], [8], [11], nor risks assessment activities (CH1.2) [30], [8], [11]. In addition, Beznosov et al. [7], Boström et al. [8], and Ge et al. [11] claim[4] that some security activities need to be repeated for each iteration because each iteration should include the full development life-cycle (CH1.3). Moreover, development iterations are of limited time, often few weeks, which makes fitting security activities (e.g., security requirements elicitation) challenging because they are often time consuming (CH1.4) [16], [3].

[3]Note that SS-CMM [15] considers the administration tasks as part of the development process.

[4]We use the term "claim" to state that authors did not necessarily justify their statements.

Incremental challenges. Security requirements are constraints on the functional requirements [13]. Beznosov and Kruchten [7] claim that code refactoring, which is a practice in the agile development methods, could break such constraints (CH2.1). They also claim that continuous code changes limits the ability to complete security assurance activities (CH2.3), a viewpoint shared by Wayrynen et al [30]. Wayrynen et al. claim also that changing the requirements and design breaks the system security requirements (CH2.2) [30]. These changes, according to Bartsch, make tracing the requirements to the security objectives (CH2.4) challenging [3].

Security assurance challenges. ASD methods advocate for light documentations. Beznosov and Kruchten [7], Woody [32], and Alnatheer et al. [1] claim that this practice conflicts with the use of documentation for security assessment (CH3.1). Beznosov and Kruchten [7] claim also that the test philosophy of agile developers (i.e., rely on tests to confirm that requirements are implemented) conflicts with security needs because tests are, in general, insufficient to ensure the implementation of security requirements (CH3.2). In addition, Wayrynen et al. [30] claim that tests, in general, do no cover all vulnerabilities cases (CH3.3) and Woody [32] claims that security tests are difficult to automate (CH3.4). Moreover, the agile development teams often improve the development processes they use to consider the lessons they learned. According to Bartsch, this good practice makes performing security audits challenging (CH3.5) because, for example, some of the activities used for the audit may be discontinued and the used resources may become not uniform [3].

Awareness and collaboration challenges. The ASD approach

TABLE IV
CAUSES OF THE CHALLENGES FOR DEVELOPING SECURE SOFTWARE USING THE AGILE APPROACH.

Code	Challenge	Agile Value	Agile principle	Security assurance practice
Software development life-cycle challenges				
CH1.1	Security requirements elicitation activity is not included in the agile development methods	-	-	-
CH1.2	Risk assessment activity is not included in the agile development methods	-	-	-
CH1.3	Security related activities need to be applied for each development iteration	-	P1, P3	S1, S3, S4, S5
CH1.4	Iteration time is limited and may not fit time consuming security activities	-	P3	S1, S3, S4, S5
Incremental development challenges				
CH2.1	Refactoring practice breaks security constraints	V4	P9	S3, S5
CH2.2	Changes of requirements and design breaks system security requirements	V4	P2	S1, S5
CH2.3	Continuous code changes makes completing the assurance activities difficult	V4	P2	S3, S5
CH2.4	Requirement changes makes the trace of the requirements to security objectives difficult	V4	P2	S5
Security assurance challenges				
CH3.1	Security assessment favors detailed documentation	V2	P7, P10	S4, S5
CH3.2	Tests are, in general, insufficient to ensure the implementation of security requirements	-	P7, P10	S5
CH3.3	Tests do no cover in general, all vulnerability cases	-	P7, P10	S5
CH3.4	Security tests are in general difficult to automate	-		-
CH3.5	Continuous changing of the development processes (to support lesson learned) conflicts with audit needs of uniform stable processes	V4	P11, P12	S5
Awareness and collaboration challenges				
CH4.1	Security requirements are often neglected	-	P7, P10	-
CH4.2	Developers lack experience on secure software	-	-	-
CH4.3	Customers lack security awareness	-	-	-
CH4.4	Developer role must be separate from security reviewer role to have objective results	V1	P4, P6	S5
Security management challenges				
CH5.1	Security activities increases the cost of the software	-	-	S1,S2,S3,S4, S5
CH5.2	There are no incentive for organizations to develop security features in early increments	-	-	-
CH5.3	Organizations compromise security activities to accommodate accelerated releasing schedule	V2	P3	S1, S3, S4, S5

Notes:
(1) Challenges that are not caused by the agile values and principles, neither by the security assurance practices are marked with color.

encourages the developers and customers to collaborate. Also, developing secure software requires the project collaborators to be educated about why and how to develop secure software. Unfortunately, according to Bartsch, agile developers require further training about developing secure software (CH4.2) [3]. He claims also that security requirements are in practice often neglected in favor of functional requirements [3](CH4.1). Wayrynen et al. [30], Woody [32], and Alnatheer et al. [1] share the viewpoint of Bartsch about CH4.2 and also propose the inclusion of security experts in the development teams to help spreading the knowledge about security. Alnatheer et al. [1] and Bartsch [3] claim also that customers lack security awareness (CH4.3), which limits their ability to help the developers develop secure software because e.g., they cannot state the security requirements for their products. In addition, Woody [32] and Konglsi [20] claim that software assessment should be performed by security experts instead of the project developers to ensure objectivity of the results. (CH4.4).

Security management challenges. Alnatheer et al. [1] claim that making software secure increases the cost of the products because it requires more development effort (CH5.1) and that organizations tend to compromise security when they have accelerated releasing schedule (CH5.3). In addition, Bartsch [3] and Woody et al. [5] claim that there are, currently, no incentive for organizations to develop security features in early increments (CH5.2).

In the next section we analyze the validity of the challenges reported above.

V. ANALYSIS OF THE CHALLENGES OF DEVELOPING SECURE SOFTWARE USING THE AGILE METHODS

This section evaluates the validity of each of the identified 20 challenges. First, we discuss how each of the agile values and principles, and the security assurance practices causes the challenges reported in the literature. We discuss next, in Subsection V-C, the (6) invalid challenges that are reported in the literature. We classify the challenges afterwards, in Subsection V-D, based on the cause categories. Table IV summarizes the evaluation.

A. Analysis of the relationship between the agile values/principles and the identified challenges

Value V1. Developers can perform security assessment to identify flaws and vulnerabilities and address them. However, the security assessment best practices favor separating the security evaluators from the developers to avoid the influence of social relations on the results, which helps to have objective assessment. This is not aligned with V1 that favors interactions between the team members (CH4.4).

Value V2. Developing secure software requires performing security assurance activities, such as static or dynamic analysis that are not critical to delivering working software (CH 5.3). In addition, the agile approach advocates for light documentation while security assurance techniques favor detailed documentation (CH3.1).

Value V3. V3 is not the cause of any of the identified challenges.

Value V4. The agile approach embraces responding to changes requested by the customers. Unfortunately, the requirements and design changes could break the security constraints of the software (CH2.1) and make it difficult to trace the requirements to the security objective of the system (CH2.4). Also, frequent code changes make completing the assurance activities difficult because the changes potentially invalidate the results of the security reviews, tests, and analysis (CH2.3). In addition, agile development methods encourage code refactoring [10] to improve the maintainability of the software. However, such changes could also break the security constraints (CH2.1). Moreover, security evaluators use the development process-related information, such as the architecture documents and the traceability of artifacts documents, in assessing the security of software. The audit techniques cannot be applied on different iterations if the development process changes. This is not aligned with V4 which tolerates process changes that potentially make the information that are used in the assessment inconsistent (CH3.5).

Principle P1. In the waterfall method, the security activities are distributed on the phases of the development lifecycle [24]. However, the priority for the team that uses an agile development method is to continuously deliver valuable software to the customer. This requires performing the security activities in each development iteration (CH1.1).

Principle P2. Though the ASD approach embraces requirement changes to fulfill late customer needs, these changes could potentially break the system security requirements (CH2.2) and make tracing of the requirements to the security objectives difficult (CH2.4). In addition requirement changes cause code changes, which could make completing security assurance activities such as code review difficult (CH2.3).

Principle P3. Delivering a *secure* working software every short period requires performing the security activities for each period. This requires fitting the security activities in the iteration period (CH1.3 and CH1.4). The short iteration duration can also lead the development teams to compromise on security activities (CH5.3); the developers may rush to deliver working software rather than secure working software.

Principles P4 and P6. Requiring that the business people and developers meet and collaborate daily throughout the project improves the social relations between them and helps to avoid misunderstandings. However, the security evaluator is somehow a judge; they should not be involved in the development process, neither have social relations with the development team to be objective in their evaluation (CH4.4).

Principles P7 and P10. The agile approach encourages doing the minimum work and evaluates the progress based on working software. This is practiced by developing only code that passes the acceptance tests that asserts that the software works. However, security requirements are negative requirements (e.g., only authorized users can access the feature) while tests check positive requirements. Thus, in general, tests are not sufficient to ensure the implementation of security requirements (CH3.2) and may not cover, in general, all the vulnerabilities cases (CH3.3). Thus, measuring the progress of a project primarily on working software is not enough, these measurements should also include the evaluation of security arguments. In addition, the agile approach favors writing a light documentation which conflicts with some security assurance needs of detailed documentation such as architecture review.

Principle P9. The continuous attention to technical excellence is practiced by code refactoring [10]. However, code refactoring can break security constraints (CH 2.1).

Principles P11 and P12. Agile development team members self-organize to improve their development tools, techniques, and processes. However, the assessor uses the development process-related information in the security evaluation. Thus, audit techniques that use such information cannot apply on different iterations if the processes change.

B. Relation of the challenges with the security assurance practices

Security assurance practice S1. System security policies are constraints on the system requirements. Requirements and design changes in the different development iterations can break the constraints (CH2.2). For example, the addition of a new feature "log access to data" could conflict with existing access control policy if the policy may not apply to the log file–which could be accessed by entities not considered in the policy. In addition, this assurance activity has to be applied for each development iteration (CH1.3), though iteration time is limited and may not fit to finalize the security activities (CH1.4). Also, specifying the security policies is time consuming and it increases the software development cost (CH5.1). This encourages the organizations to compromise the assurance activity to accommodate accelerated releasing schedule (CH5.3).

Security assurance practice S2. Developing secure software requires training the developers, which increases the cost of software development (CH5.1).

Security assurance practice S3. Identifying and categorizing the flaws of secure software must be performed in each iteration (CH1.3), which costs time (CH1.4). Moreover, code changes, even code refactoring, may introduce code vulnerabilities (CH2.1)(CH2.3) and may contribute to increasing the software development cost (CH5.1). It may also delay the development, which leads organizations to compromise the assurance activity (CH5.3).

Security assurance practice S4. This assurance practice has to be applied to each development iteration (CH1.3), though iteration time is limited and may not be sufficient to finish the activity (CH1.4). In addition, security risk assessment is often based on detailed documentation of the software, which contradicts with the agile practice of having light documentation (CH3.1). Moreover, security risk assessment is a time consuming activity and it increases the software development cost (CH5.1). This could cause a delay in the releasing

schedule, which encourage organizations to compromise the activity (CH5.3).

Security assurance practice S5. The security verification and validation gate asserts whether software is secure, with respect to its security requirements, or not (CH3.1). This should be done in every iteration (CH1.3), which is challenging due to time limitation (CH1.4). Moreover, this activity does not add value to the functionalities of the software and it increases its cost (CH5.3). The results of the verification and validation could be invalidated by code refactoring (CH2.1) and also by requirements and design changes (CH2.2). Furthermore, frequent requirement changes can make tracing their impacts on the security objectives, a technique used in security verification, challenging (CH2.4)(CH3.5). We note also that the common practices in security assurance rely on independent evaluators, not on the developers of the software (CH4.4); and that tests are, in general, insufficient to ensure the implementation of security requirements (CH3.2) and may not cover all vulnerability cases (CH3.3).

C. Challenges which have no relations to ASD values or principles neither to the security assurance practices

The inclusion of security requirement activity (CH1.1) is independent of the development method; the agile values and principles do not prevent performing such activity as part of the development life cycle. This also applies to the risk assessment activity (CH1.2). For instance, in Scrum a risk management activity could be included in the backlog and the developers could monitor the risk exposure of their software using the risk burndown chart. Also, the problems of automating security tests (CH3.4), developers lack of experience on secure software (CH4.2), and customers' lack of security awareness (CH4.3) apply to both the agile and waterfall development methods. In addition, the lack of incentives for organizations to develop security features in early increment (CH5.2) is not influenced by the agile values or principles; it applies to both the agile and the waterfall development methods.

D. Summary

The analysis reveals that not all of the challenges of developing secure software discussed in the literature are valid. They are of 4 categories.

1. Challenges that are caused by the agile values or agile principles and security assurance practices. This category includes 12 challenges: CH1.3, CH1.4, CH2.1, CH2.2, CH2.3, CH2.4, CH3.1, CH3.2, CH3.3, CH3.5, CH4.4, and CH5.3. We observe that all the incremental development challenges and most of the security assurance challenges belong to this class. In our opinion this category of challenges needs the most attention.

2. Challenges that are caused by the agile values or principles. This category includes only 1 challenge: CH4.1.

3. Challenges that are caused by the security assurance practices. This category includes 1 challenge: CH5.1.

4. Challenges that are not caused by the agile values or principles nor the security assurance practices. This category includes 6 claimed challenges: CH1.1, CH1.2, CH3.4, CH4.2, CH4.3, CH5.2. These challenges are not valid; they do not make developing secure software using the agile methods more difficult than using e.g., the waterfall method.

VI. Limitations and impacts of the study

A. Impacts of the study results

This study provides a review of the literature about the challenges associated with developing secure software using the agile approach and the agile methods. It potentially impacts the research and practice of developing secure software using the agile methods [19]. First, the study makes it easier–for especially new researchers–to get an overview about the domain, understand the literature, and construct research questions. Second, it could be used to identify relevant literature for the "related work" section of primary studies. Third, it justifies the need for further research to address the challenges.

B. Limitations of the study

The main limitations of this study are: the possibility to miss relevant publications, the bias in the selection of the relevant studies, the inaccuracy of the data extraction, subjectivity of the evaluation of the challenges, and the low number of relevant publications. The first three limitations are common to systematic literature review [9], [17].

Missing relevant publications. We used in this study a set of keywords to search for potentially relevant papers. This approach misses the papers that are not indexed by the search engine we used and also the papers that are not indexed with the keywords we choose. We note that keywords are both discipline and language specific and are not standardized [17]. In addition, the results of search engines are not consistent. For instance, we searched in June 2015 the ACM library. We obtained 1601 records when we queried the library from Fraunhofer SIT and 543 when queried it from TU Darmstadt, Germany. Fortunate, the issue did not change the set of relevant papers of our study.

Bias in the selection of the relevant studies. We selected potentially relevant papers from the papers identified in the search steps based on the titles and abstracts. This approach discards the papers that do not "indicate" discussion about the challenges of developing secure software using the agile approach. Thus, we may have missed relevant papers.

Inaccuracy of the data extraction. We read the selected papers and extracted the challenges reported in section IV. Some challenges may have been missed because their discussion in the related papers was confusing and lack transparency. Two of the authors read the papers and discussed the challenges that each identified, which helps to control the inaccuracy of the data extraction.

Subjectivity of the evaluation. This limitations is due to the fact that the evaluation is based on human perception of the challenges and evaluation criteria. Two of the authors

evaluated the challenges and discussed their disagreement, which should reduce the subjectivity of the evaluation.

Low number of relevant publications. The number of relevant papers, 10, is low. This is because the topic is "new."

VII. Conclusion

There has been an ongoing discussion about the difficulties to use the agile development methods to develop secure software. Among the argumentation is that agile development methods embrace requirement changes and frequent deliveries while developing secure software requires the use of verification gates and refinement of artifacts. This paper reports about a review of the literature to identify the challenges reported in the literature about developing secure software using the agile approach and methods. It summarizes the challenges discussed in the literature and evaluates the relations of each to the agile values, the agile principles, and the security assurance practices. This results to the identification of 20 challenges found in 10 publications, 14 of the 20 are deemed valid and 6 are found to be not caused by the agile values, the agile principles, nor the security assurance practices.

The results of this study are an evidence that developing secure software using the agile methods is challenging, which calls for research to address the challenges. We are working on contributing to adapt the agile development methods and the security assurance practices to enable developing secure software using the agile methods smoothly.

References

[1] A. Alnatheer, A. M. Gravell, and D. Argles, "Agile security issues: A research study," accessed on Apr. 2015. [Online]. Available: http://esem2010.case.unibz.it/idoese/docs/alnatheer.pdf

[2] S. W. Ambler. (2009, Dec.) The agile scaling model (ASM): adapting agile methods for complex environments. IBM. [Online]. Available: ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/raw14204usen/RAW14204USEN.PDF

[3] S. Bartsch, "Practitioners' perspectives on security in agile development," in *Proc. the Sixth International Conference on Availability, Reliability and Security*, ser. ARES '11, Vienna, Austria, Aug. 2011, pp. 479–484.

[4] M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," http://agilemanifesto.org/, accessed on Apr. 2015.

[5] S. Bellomo and C. C. Woody, "Dod information assurance and agile: Challenges and recommendations gathered through interviews with agile program managers and dod accreditation reviewers," Carnegie Mellon University, Tech. Rep. CMU/SEI-2012-TN-024, Nov. 2012. [Online]. Available: repository.cmu.edu/cgi/viewcontent.cgi?article=1674&context=sei

[6] L. ben Othmane, P. Angin, H. Weffers, and B. Bhargava, "Extending the agile development approach to develop acceptably secure software," *IEEE Transactions on Dependable and Secure ComputingDependable and Secure Computing*, vol. 11, no. 6, pp. 497 – 509, Nov. 2014.

[7] K. Beznosov and P. Kruchten, "Towards agile security assurance," in *Proc. of the 2004 Workshop on New Security Paradigms*, ser. NSPW '04, White Point Beach Resort, Canada, 2004, pp. 47–54.

[8] G. Boström, J. Wäyrynen, M. Bodén, K. Beznosov, and P. Kruchten, "Extending XP practices to support security requirements engineering," in *Proc. of the 2006 international workshop on Software engineering for secure systems*, Shanghai, China, May 2006, pp. 11–18.

[9] T. Dyba and T. Dingsoyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9-10, pp. 833 – 859, Aug. 2008.

[10] M. Fowler and K. Beck, *Review: Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[11] X. Ge, R. F. Paige, F. A. Polack, H. Chivers, and P. J. Brooke, "Agile development of secure web applications," in *Proc. of the 6th International Conference on Web Engineering*, ser. ICWE '06, Palo Alto, CA, July 2006, pp. 305–312.

[12] K. M. Goertzel, T. Winograd, H. L. McKinley, P. Holley, and B. A. Hamilton, "Security in the software lifecycle," Online, August 2006, draft version 1.2. [Online]. Available: www.cert.org/books/secureswe/SecuritySL.pdf

[13] C. B. Haley, R. Laney, J. D. Moffett, and B. Nuseibeh, *Integrating Security and Software Engineering: Advances and Future Visions*. Hershey, PA: Idea Group Publishing, 2007, ch. Arguing Satisfaction of Security Requirements, pp. 16–43.

[14] M. Jazayeri, "Some trends in web application development," in *Proc. Future of Software Engineering*, ser. FOSE '07, Washington, DC, USA, May 2007, pp. 199–213.

[15] JTC1 Information technology, committee SC27, *Information technology –Security techniques – Systems Security Engineering – Capability Maturity Model (SSE-CMM)*, International Organization for Standardization (OSI) Std. ISO/IEC 21 827, 2008. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=44716

[16] H. Keramati and S.-H. Mirian-Hosseinabadi, "Integrating software development security activities with agile methodologies," in *Proc. IEEE/ACS International Conference on Computer Systems and Applications*, Doha, Qatar, Apr. 2008, pp. 749 –754.

[17] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a tertiary study," *Information and Software Technology*, vol. 52, no. 8, pp. 792–805, Aug. 2010.

[18] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," University of Durham, Durham, UK, Tech. Rep. EBSE-2007-01, July 2007.

[19] B. A. Kitchenham, D. Budgen, and O. P. Brereton, "Using mapping studies as the basis for further research a participant-observer case study," *Information and Software Technology*, vol. 53, no. 6, pp. 638 – 651, 2011.

[20] V. Kongsli, "Towards agile security in web applications," in *Proc. 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, Portland, OR, USA, Oct. 2006, pp. 805–808.

[21] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *Computer*, vol. 36, no. 6, pp. 47–56, Jun. 2003.

[22] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2003.

[23] L. M. Maruping, V. Venkatesh, and R. Agarwal, "A control theory perspective on agile methodology use and changing user requirements," *Info. Sys. Research*, vol. 20, no. 3, pp. 377–399, Sep. 2009.

[24] G. McGraw, *Software Security: Building Security In*, ser. Addison-Wesley Software Security Series. Addison-Wesley, 2006.

[25] ——, "On bricks and walls: Why building secure software is hard," *Computers & Security*, vol. 21, no. 3, pp. 229–238, 2002.

[26] S. R. Palmer and J. M. Felsing, *A practical guide to feature-driven development*, 1st ed. Prentice Hall, Feb. 2002.

[27] J. Saldana, *The coding manual for qualitative researchers*, 1st ed. London, UK: SAGE Publications Ltd, 2009.

[28] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[29] B. Sullivan, "Agile security; or, how to defend applications with five-day-long release cycles," Black Hat, DC,, Jan. 2010. [Online]. Available: http://www.blackhat.com/presentations/bh-dc-10/Sullivan_Bryan/BlackHat-DC-2010-Sullivan-SDL-Agile-slides.pdf

[30] J. Wayrynen, M. Boden, and G. Bostrom, "Security engineering and extreme programming: An impossible marriage?" in *Proc. 4th Conference on Extreme Programming and Agile Methods*. Calgary, Canada: Springer, Aug. 2004, pp. 117–128.

[31] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering*. Berlin Heidelberg: Springer-Verlag, 2012.

[32] C. Woody, "Agile security - review of current research and pilot usage," Carnegie Mellon University, Tech. Rep., Nov. 2013. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=70232