Copyright ⓒ 2014 KSII

Integrating Software Security into Agile-Scrum Method

Imran Ghani[1], Zulkarnain Azham[1], and Seung Ryul Jeong[2]
[1]Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia
[e-mail: imran@utm.my, zulkarnain.azham@gmail.com]
[2]Graduate School of Business IT, Kookmin University, Korea
[e-mail: srjeong@kookmin.ac.kr]
*Corresponding Author: Seung Ryul Jeong

Received November 23, 2013; accepted January 21, 2014; published February 28, 2014

Abstract

Scrum is one of the most popular and efficient agile development methods. However, like other agile methods such as Extreme Programming (XP), Feature Driven Development (FDD), and the Dynamic Systems Development Method (DSDM), Scrum has been criticized because of lack of support to develop secure software. Thus, in 2011, we published research proposing the idea of a security backlog (SB). This paper represents the continuation of our previous research, with a focus on the evaluation in industry-based case study. Our findings highlight an improved agility in Scrum after the integration of SB. Furthermore, secure software can be developed quickly, even in situations involving requirement changes of software. Based on our experimental findings, we noticed that, when integrating SB, it is quite feasible to develop secure software using an agile Scrum model.

Key Words: Scrum, Software Security, Agile Methodologies, Security Backlog

A preliminary version of this paper was presented at APIC-IST 2013 and was selected as an outstanding paper.

http://dx.doi.org/10.3837/tiis.2014.02.0019

1. Introduction and Problem Background

Agile methodologies such as Scrum, Extreme Programming (XP), Feature Driven Development (FDD), and the Dynamic Systems Development Method (DSDM), have gained enough attention for software development in recent years [1]. A number of organizations [2,3] that practice agile methods suggest that agile methods help during the software development process by emphasizing rapid development statements [4,5]. In fact, agile methods are more flexible and help to reduce risks of project failure. However, they need to follow several rules related to the agile manifesto, including those concerning less documentation and team member interactions, which provide for appropriate communication with customers and other users. However, some researchers and practitioners [4,6,7] highlighted a critical software problem – software security [8,10] and [11]. As mentioned in the abstract, like other agile methods, the Scrum model does not provide guidelines for dealing with the security aspects of software [12]. The original Scrum model does not include software security planning from the start [13]. This means that security practices cannot be implemented by the Scrum team. As a result, the agile team may produce vulnerable software that can be exploited by attackers [8]. On the contrary, if the Scrum agile team concentrates on solving software security issues, the agility of the Scrum process may be affected [14].

Thus, this study attempts to discover whether or not security practices such as SB can work with the Scrum agile model. If so, then developers who focus on developing secure software using Scrum may apply the proposed SB.

1.1. Degree of Agility

Based on a survey and assessment of various contemporary definitions, Qumer and Henderson-Seller [15] offer the following definition for the agility of any entity (www.agilemanifesto.org):

"*Agility is a persistent behavior or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment*."

The aforementioned definition can then be simplified by a researcher into five key elements of agility that represent flexibility, speed, leanness, learning and responsiveness. To validate the performance of an agile model, the degree of agility needs to be evaluated by measuring the following elements.

- Flexibility: The agile method should be flexible enough to welcome a change in requirements during any phase. A lack of flexibility will create a serious crisis in the agile software process. In other words, we can say that this represents the ability to respond to any change at any given time.
- Speed: As the speed of software delivery is one of the most important elements in the agile manifesto, it needs to be considered as an element of agility [16].
- Leanness: This represents the elimination of waste or the doing of more with less. Through maximizing the utilization of all resources, and the elimination of unnecessary resources, all tasks are streamlined. At the same time, however, the level of quality should be maintained [17].
- Learning: Focuses on improvement during and after product development when software had been delivered to the client.

- Responsiveness: This means responding to any change, either within the team, or in the requirements of the software itself [15].

Any software development process that implements the whole of agility, as stated above, can be considered to be an agile method. However, its degree of agility needs to be evaluated before it can be considered to be a suitable agile model.

2. Literature Review

2.1. Security Issues and Agility Concerns

In general, existing security standards and practices do not easily align with the agile manifesto due to the strictness of the security requirements phase. In fact, the security practices that are still widely followed were developed before the agile manifesto was introduced. The current reality is that modern software projects undergo constant changes, which leads to process re-iteration, or in other words, repetition in the development process. This new phenomenon creates a dilemma in that there are no established security standards that consider agility, which limits the potential of any agile model, including those that use Scrum [7,18].

2.2. Security Issues and Scrum

In particular, Scrum itself has a number of limitations related to security. These are stated below.

- As the Scrum release cycle is too short, there is not enough time for the development team to address security requirements for each release.
- Although the inclusion of security elements in the existing Scrum does not present a significant issue, issues do arise when such inclusions affect the process characteristics or agility of Scrum.
- The authors, Kevin Brady [19] and Mikkosiponen *et al.* [20], state that knowledge monopolies in a Scrum team are the main factors causing an absence of documentation during the requirement planning. This happens due to the team players not having enough skill in regards to security.
- The lack of guidelines in the collection of security requirements is also a factor, as there is no requirements freeze in agile as there is in the other conventional SDLCs [7].
- One of the research groups, known as Securosis [22], found that, although the project manager played a major role in Scrum, their lack of security awareness and pressure to complete the project within a minimal amount of time affected the entire process. For example, threat analysis needed to be included in the requirements capturing/analysis guidelines. However, it was either not included at all or the Scrum Product Owner (PO) could not complete it due to a lack of time [22]. The same issue was faced during security testing in sprint phases. As a result, the agility of the process was affected [21].

In order to overcome these issues, this study attempts to suggest ways in which the Scrum process can be suitably combined with security practices. Previous research [23] proposed a solution for this situation. However, there was not enough data regarding how the proposed solution affected Scrum agility. In this paper, we will present our findings regarding these issues.

2.3. Software Security Principles

Recapping Security Principles. Security principles are the primary methods used to determine methods for securing software. They are universal guidelines adopted by every developer and project planner, and have been adopted by practitioners and experts in the security field in order to mitigate risks in the architecture, design, implementation, testing and maintenance phases. As illustrated in **Table 1**, security principles act as a guideline for a number of existing systems and approaches. As shown in the table, much research has been proposed showing solutions that satisfy some or all of the security principles. The 'X' symbol is used when researchers have not mentioned the security principle, while the'√' symbol is used when it has been mentioned.

2.4. Software Security Engineering in Phases

Security activities generally include three phases. These are known as the requirement, development and testing phases. A software engineer can sabotage the software at any point in its development life cycle through the usage of intentional exclusions from, or inclusions in. Sabotage can also be performed by modifying the requirements specification, the threat models, the design documents, the source code, the assembly and integration framework, the test cases and test results, or the installation and configuration instructions and tools. The secure development practices described in this book are, in part, designed to help reduce the exposure of software to insider threats during the development process. For more information on this aspect, see "Insider Threats in the SDLC" [13].

2.4.1. Security during requirement phase: There are a few existing techniques that can be used for eliciting the security requirements. Some of these have been created with attention paid to security, while others have been created in outdated requirement engineering style. These can be enhanced to add a security element. Existing techniques for the security requirement include Misuse cases [34], Soft Systems Methodology [25], Quality Function Deployment [26], Controlled Requirements Expression [35], Issue-based information systems [28], Joint Application Development [29], Feature-oriented domain analysis [36], Critical discourse analysis [30] and the Accelerated Requirements Method [31]. It is impossible to create secure software without keeping security in mind from the beginning. Preparation must be started during the requirement phase because at the development phase, developers simply follow what is stated in the requirements. For the scrum method, there is an existing security method proposed by EAST Methodology [33], which is integrated in Scrum. In this method, we can see there is no special backlog for the security part. Instead, the security analysis is combined in the product backlog. In Scrum, the product backlog should not be too complex, as this will affect the leanness agility.

Table 1. Security Principles

	Least privilege	Failing securely	Securing the weakest link	Defence in depth	Separation of privilege	Economy of mechanism	Least common mechanism	Reluctance to trust	Never assuming that your secrets are safe	complete mediation	Psychological acceptability	Promoting privacy	Design Principles
Julia H. Allen et. al., 2009	√	√	√	√	√	√	√	√	√	√	√	√	X
Us-Cert	√	√	√	√	√	√	√	√	√	√	√	√	√
Open Web Application Security Project (OWASP)	X	√	X	√	X	√	X	√	X	X	√	X	√
Gary McGraw and John Viega, 2009	√	√	√	√	√	√	√	√	√	√	√	√	√
NIST	√	√	√	√	√	√	√	√	√	√	√	√	√
SANS	√	√	√	√	√	√	√	√	√	√	√	√	√
IBM	√	√	√	√	√	√	√	√	√	√	√	√	√
Saltzer and Schroeder	√	√	X	X	√	√	√	X	X	√	√	X	√

2.4.2. Existing Secure Software Development Lifecycle: Microsoft has developed a guide for the SDL agile process. However, it is commonly believed that because SDL was originally created for Microsoft's big showcase box products such as Windows and SQL Server, it only works for those kinds of products. This is, of course, patently false. Virtually every Microsoft product and online service, large or small, follows the SDL. Many other organizations outside of Microsoft are also successfully implementing the SDL. However, while the content of the SDL and its requirements and recommendations may be universal, the structure of the SDL as originally designed is more suited to long running waterfall or spiral style development methodologies [32].

Table 2. Analysis of Existing Models Secure Software Development

Phases	**Model**	**Activity**	**Security Principles**
Requirements	• Misuse cases [34] • Quality Function Deployment [26] • Controlled Requirements Expression [27] • Issue-based information systems [28] • Joint Application Development [29] • Feature-oriented domain analysis [36] • Critical discourse analysis [30] • Accelerated Requirements Method [31].	Analysis of the possible threat	Not Mentioned
Development	Microsoft Security Development Lifecycle for Agile Development version 1.0	Adds security requirement at all sprint. Security Testing.	Not Mentioned

Testing	Extended Agile Security Testing(EAST)	Adds security requirement. Documentation. Security Testing.	Not mentioned but the activity can be representing the security principle needed.

The table above shows that the existing agile models that are used to develop secure software do not clearly mention or integrate the security principles in the requirement, development or testing phases. The X symbol means the corresponding model does not solve the issue and provide a solution, while the√ symbol means that it does solve the issue and provide a solution.

Table 3 shows an analysis of existing secure software development which compares the Microsoft and EAST models. This analysis has been obtained from a recent literature review (2005 - 2010).

In order to address some of the highlighted issues, we previously proposed the idea of SB for Scrum security practices [23]. For the sake of the reader, we will illustrate our previously proposed SB in the next section. However, the main objective of this paper is to analyze the effectiveness of the SB and provide insights obtained from an industry-based case study.

Table 3. Analysis of Existing Secure Software Development Lifecycle

Sources	**Issues**	**Models**		**Proposed Solutions**
		MS	**EAST**	
[32]	Traditional Security Development Life Cycle is does not fit with Agile.	√	√	Proposes an enhancement existing Agile security life cycle.
[37]	In order to overcome the security aspect in SDLC like missing a security phase in scrum, proposes to integrate with another development methodologies.	X	X	Imports some practices; example, pair programming from eXtreme Programming.
[7]	Security must be considered from the start of a development process, and it is “anti-freeze requirement”.	X	√	Create a security activity from the beginning
[38]	To add the threat analysis part in planning sprint is not suitable because it is too large effort to squeeze into sprint session.	X	√	Threat analysis from sprint
[22]	The average sprint duration of two weeks is simply too short for security test.	X	X	Suggests security testing for only selected features.
[19, 20]	Knowledge monopolies of security staff due to the absence of documentation for security part.	√	X	Make a comprehensive documentation but simple based on security principle.
[38]	“Inject” certain security related activities into the scrum.		√	Creates a security activities
[14]	Need to integrate a security activity with all agile activities.		√	Suggests security activities in all phases.

3. Analysis of Proposed Security Backlog (SB)

3.1. Validation and Evaluation

In this section our previously proposed SB and its integration into the Scrum model is analyzed and the authors explain the overall enhanced Scrum model.

As mentioned in **Table 4**, existing frameworks do not offer a solution for the security issues in Scrum. Therefore, our aim is to merge two new elements into the existing Scrum. These are 1) a new security component known as SB and 2) a new role known as Security Master (SM). The SM is in charge of the SB during the Scrum lifecycle.

Table 4. Security Description in Each Phase [23]

Product Backlog	The Product Backlog is the master list of all functionality desired in the product. When using Scrum, it is not necessary to start a project with a lengthy, upfront effort to document all requirements. The documentation for security will be absence here because of the team want produce the simple documentation.
Release Backlog	A release backlog is a subset of the product backlog that is planned to be delivered in the coming release. Partition the product backlog a release backlog for each release. The release backlog is the subset of product requirements that will deliver in a given release. The prioritize feature that want to be release quickly will in a danger if not analyze it with the threat analysis phase.
Sprint Backlog	The sprint backlog is the list of tasks that the Scrum team is committing that they will complete in the current sprint. Here, the time release has been decided, and any addition in security part, will be hard to manage in time.
Testing and shippable phase	Here all product has been integrated and will be testing for a final. When it comes to this phase and the security part will cost a lot of time and money here to maintain.

The √ symbol means that the phases should practice the respective security principle, while the X symbol means that phases do not need the respective security principle. Based on analysis in **Table 3** and security principle roles in **Table 2** introduced by various researchers, we come to a conclusion in **Table 5**. **Table 5** shows analysis regarding how security guidelines should be aligned to each related backlog and phase. The chosen factor is based on security principles stating that nearly all principles can provide guidelines for requirement, development and testing activities.

Table 5. Relation between Phases and Security Principle

Security Principle / Phase	Least privilege	Failing securely	Securing the weakest link	Defence in depth	Separation of privilege	Economy of mechanism	Least common mechanism	Reluctance to trust	Never assuming that your secrets are safe	complete mediation	Psychological acceptability	Promoting privacy	Design Principles
Product Backlog	√	√	√	√	√	√	√	√	√	√	√	√	√
Release Backlog	√	√	√	√	√	√	√	X	X	√	X	√	X
Sprint Backlog	√	X	√	√	√	√	√	X	X	√	√	√	**X**
Testing and shippable phase	√	√	√	√	√	√	√	√	√	√	√	√	√

The enhanced Scrum model in **Fig. 1** has two additional components. These are:

1. A Security Backlog (SB) that manages security in Scrum. (please refer to **Fig. 2**)
2. A Security Master (SM)that handles security in Scrumand is in-charge ofthe SB.

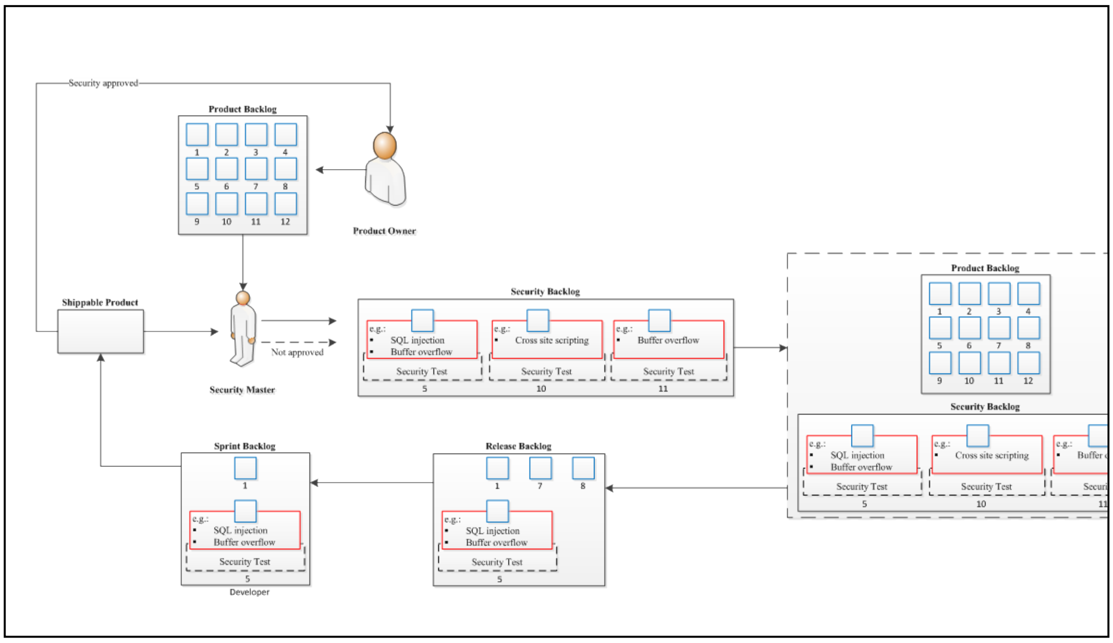

Fig. 1. Enhanced Scrum process with additional SB and SM

As illustrated in **Fig. 1**, the process flow is shown below.

1. The requirements are translated into product backlog and go through SB.
2. Here, the SM figures out certain features in the product backlog that require security attention.
3. The Security Master marks (dotted in illustrations) the selected features in the security backlog. The SM creates a document of its activity for use as a reference during the

development and testing phases. The marked security concerns are noted in the sprint backlog for the attention of the developers.

4. Testing is verified in sprint phases by the security master.

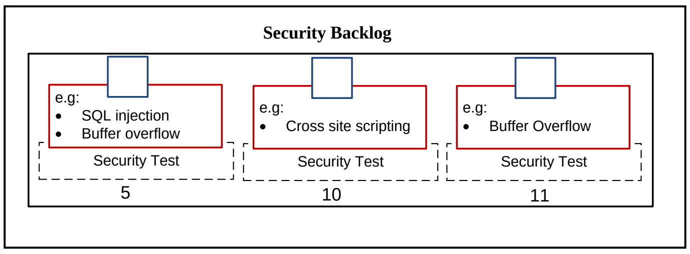

Fig. 2. Process Security Backlog in Scrum

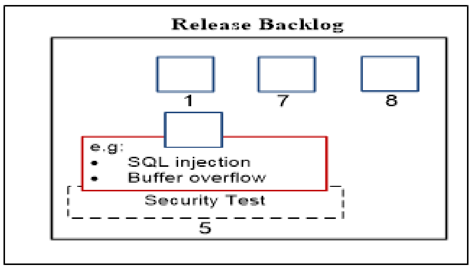

Fig. 3. Release Backlog Highlighted with Security Related Features

As shown above, **Fig. 3** displays SQL injection and buffer overflow attack possibilities assigned by the SM. Since the agile method documentation needs to be as simple as that suggested in The Systems Security Engineering Capability Maturity Model (SSE-CMM) and CC, the SB also helps to produce a simple but comprehensive document. At the same time, it attempts to provide clear information regarding the security risks.

4. Evaluation

4.1. Evaluation of agility in Each Phase

The three phases involved in this Scrum enhancement model are the requirement phase, development phase and testing phase. As discussed in the beginning, the agility is the variable to be evaluated for each phase.

4.1.1. Requirement Phase and Degree of Agility. The requirement phase is conducted by the Product Owner. After collecting the requirements, the selected features are populated in the Product Backlog (hereafter, PB). The PB is given to the Security Master (SM) to evaluate and then to the SB. The security backlog is evaluated using this experience and a conclusion is reached based on the agility provided. If agility is affected, the product owner (PO), Scrum Master and SM suggest improvements.

- Speed: Overall, speed does not affect the delivery deadline when a security backlog is implemented in the requirement phase. Since the SM has an additional workforce available, they can identify if previously developed security libraries/classes can be reused. This information may be provided in the description of security requirements in the SB. Another way in which the SB can save time for the PO is by having guidelines without the need for research, or by finding a specific security requirement. Since the PO and SM have simple guidelines to follow in terms of security, the results of security requirements can be generated quickly. In conclusion, the security backlog would not affect the overall delivery deadline. It can also save time and produce swift security results.

- Flexibility: In short, flexibility refers to adaptability to expected change requirements. In the beginning, it is difficult to be flexible due to the lack of information within the security backlog. However, this information is also simple enough to analyze. Although flexibility can be affected in the beginning, with time it can be recovered.
- Leanness: The security backlog can complete a task within the shortest available time span because of its simplicity. Due to this simplicity, the security backlog can provide a proper understanding to the PO in a short amount of time. As a result, leanness will not be affected.
- Learning: If the description in a security backlog is user friendly, it can maintain or improve upon the current knowledge of the PO. Since the history of previous requirements can be maintained inside the SB, it can also provide experience to a new PO. As a result, the security backlog in the requirement phase will have a positive learning curve.

4.1.2. Development Phase and Degree of Agility. The development phase is evaluated by the SM and system developers in the team. The security backlog illustrated in figure 5.2 is evaluated using their experience. If the agility is affected, they suggest improvements.

- Speed: The security backlog does not cause any slowdown in developing the system within the required timeframe. However, sometimes it depends on the flexibility or the complexity of a business case. Referring to the description provided in the SB can also help the developers save time in tracing any errors or bugs that have occurred in the sub-modules. By using the SB, the developers can also add or modify any new code to any of the sub-modules for implementation of security. As a result, the speed is generally not affected.
- Flexibility: The developer can accommodate expected changes or user requirements within any sub modules during the development period. For unexpected changes the SB can be updated from time to time.
- Leanness: Since the backlog will have guidelines and standards for the developers, they can still maintain simplified codes during development to produce quality products through the help of SB. Developers can avoid adding any un-used or useless codes when developing by using the SB. This can be done in the beginning and can then be reused.
- Learning: Security backlogs can maintain or improve the current knowledge of the developers regarding the product or system that is being developed. Since the descriptions can help new developers develop new code, the security backlogs also provide experience to developers regarding the previously developed products.

4.1.3. Testing Phase and Degree of Agility. The testing phase is evaluated by the team's tester. The SB, as illustrated in, is consulted due to their experience. If affected is agility, the tester suggests improvements.

- Speed: The implementation of SB in Scrum development improves the tester's work flow. Use of the security backlog can save time during the testing phase in selecting the appropriate method, since the requirements will be focused on testing SQL injection, buffer flow, and so on. The testing process will become faster and easier as the tester comes to more quickly understand the features "to be tested" from the backlog. With this information provided in the SB, all testing can produce quick results.
- Flexibility: The SB is not rigid and offers the flexibility to add, modify or delete the security features scheduled to be tested.
- Leanness: The SB helps the tester complete tasks and activities in the shortest amount of time by going through the major list.
- Learning: The tester gains new knowledge as information inside the security backlog is stored and updated. This saved experience will allow a new tester to improve and learn more quickly.

4.2. Scrum Master's Evaluation about Responsiveness

The Scrum Master, or project manager, is the one responsible for evaluating the overall responsiveness of an agile development process using SB. It has been noted that SB did not affect the responsiveness of the overall project. SB is sensitive and can adapt to the environment. It has been applied on a small scale and accepted by an experienced team using the Scrum method. Their feedback is shown in **Table 6**.

Table 6. Industrial Feedback

Agility		**Team Role**			
		SM	**PO**	**Developer**	**Tester**
Speed	SW delivery on time	-	YES	YES	YES
	Help saving time	-	YES	YES	YES
	Produce result	-	YES	YES	YES
Flexibility	Method accommodate expected changes	-	YES	YES	YES
	Method accommodate unexpected changes	-	YES	YES	YES
Leanness	Shortest time span	-	YES	YES	YES
	Economical	-	YES	YES	YES
	Simple and quality production	-	YES	YES	YES
Learning	Update prior knowledge	-	YES	YES	YES
	Provides experience	-	YES	YES	YES
Responsiveness	Sensitiveness	YES	-	-	-
	Adaptability in team,	YES	-	-	-

Based on the industry feedback shown in **Table 6**, it is safe to say that SB can be implemented as part of the enhancement process in a Scrum method selected for security. This information has been used to calculate agility using the 4-DAT framework, in order to compare agility both before and after security backlog implementation was performed according to our model.

4.2.1. Evaluating Degree of Agility.

In order to evaluate the degree of agility, the 4-DAT framework has been used in our case study. **Table 7** shows the degree of agility using Scrum without implementing the SB. In reference to the phases in **Table 7**:

- The 0 in column four means leanness was not achieved by the Scrum master. This means the PB was not effective enough to practice security for the Scrum master.
- The 1 in other columns means that the other agility features were achieved.
- 7/7 is the total of the values in columns 1, 2, 4, 5. It means the relevant agilities in the columns were achieved.
- 0/7 is the total of the values in column 3. It means the relevant agility in the column was achieved.

Table 7. Degree of Agility before Enhancement

Scrum	Agility Features					
	Flexibility	Speed	Leanness	Learning	Responsiveness	Total
Phases						
Pre-Game	1	1	0	1	1	4
Development	1	1	0	1	1	4
Post-Game	0	1	0	0	0	1
Total	2	3	0	2	2	9
Degree of Agility	2/3	3/3	0/3	2/3	2/3	(2+3+0+2+3)/(3*5)
Practices						
Scrum Master	1	1	0	1	1	4
Scrum Teams	1	1	0	1	1	4
Product Backlog	1	1	0	1	1	4
Sprint	1	1	0	1	1	4
Sprint planning meeting	1	1	0	1	1	4
Daily scrum meeting	1	1	0	1	1	4
Sprint review	1	1	0	1	1	4
Total	7	7	0	7	7	28
Degree of Agility	7/7	7/7	0/7	7/7	7/7	28/(7*5)

In order to calculate agility the following formula has been adopted:

$$\frac{\text{Overall SUM of 1 for each Agility Feature in Each Practice}}{\text{Number of Agility Features * Number of Practices}} = \text{Degree of Agility Practices} \quad (1)$$

Thus, Degree of Agility of Practices = 28/(7*5) = 0.80. The formula, label 0 and 1 were also implemented in **Tables 7 and 8** using the same technique.

Table 7 shows the difference from **Table 8**, which includes the security backlog. The industrial experts who evaluated the SB in **Table 7** agree that the SB can be implemented and does not affect the five criteria of agility i.e., flexibility, speed, leanness, learning and responsiveness. Based on the conclusion of **Table 7**'s industrial evaluation, the security backlog considered in **Table 8** can be performed. Thus, number 1 should be put inside the blank space for all agility.

Table 8. Degree of Agility after Enhancement

Scrum	Agility Features					
	Flexibility	Speed	Leanness	Learning	Responsiveness	Total
Phases						
Pre-Game	1	1	0	1	1	4
Development	1	1	0	1	1	4
Post-Game	0	1	0	0	0	1
Total	2	3	0	2	2	9
Degree of Agility	2/3	3/3	0/3	2/3	2/3	9/(3*5)
Practices						
Scrum Master	1	1	0	1	1	4
Scrum Teams	1	1	0	1	1	4
Product Backlog	1	1	0	1	1	4
Security Backlog	1	1	1	1	1	5
Sprint	1	1	0	1	1	4
Sprint planning meeting	1	1	0	1	1	4
Daily scrum meeting	1	1	0	1	1	4
Sprint review	1	1	0	1	1	4
Total	8	8	1	8	8	33
Degree of Agility	8/8	8/8	1/8	8/8	8/8	33/(8*5)

The data in **Table 9** refers to the activity in practices for Scrum practices. The degree of agility has improved from 0.80 before implementation to 0.83 after implementation.

Table 9. Calculation Degree of Agility

Process and Practices	Scrum (Before implementation)	Scrum (After implementation)
Phases	9/15 = 0.60	9/15 = 0.60
Practices	28/35=0.80	33/40 = 0.83

The degree of agility is illustrated in Figure 4 adds a comparison before and after the security backlog has been added. The phases and practices have been calculated, and the result in **Fig. 4** shows the practices clearly.

Based on **Fig. 4**, the degree of agility has improved in the practices from 0.8 (before implementation) to 0.83 (after implementation). The improvement shown after implementation shows that the security backlog does not add any delay to speed, flexibility, leanness, learning or responsiveness if the security applied is part of the Scrum method. This shows that such implementation is relevant and can be performed without any fear of affecting agility negatively.

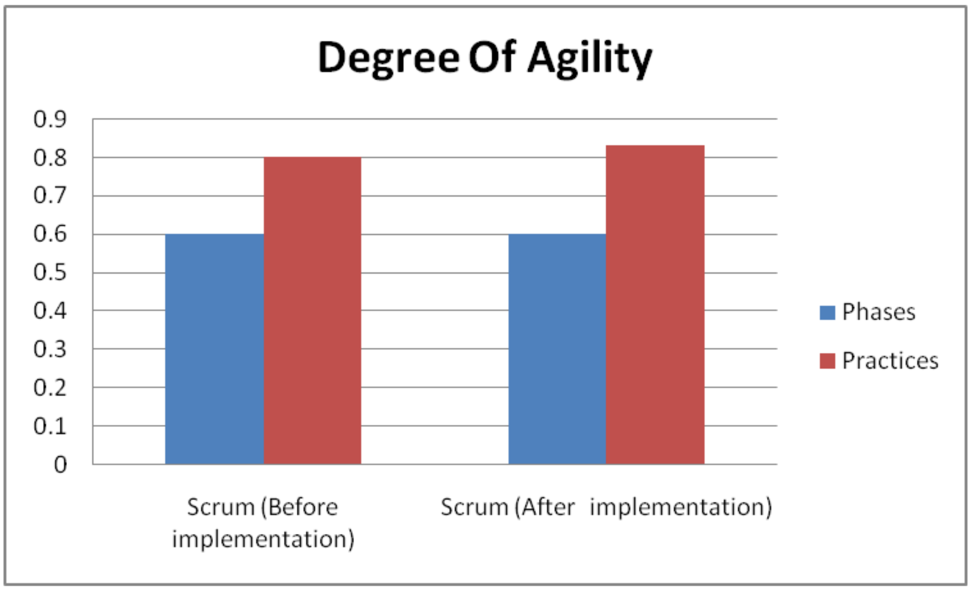

Fig. 4. Comparison Degree of Agility

5. Conclusion

After preliminary analysis, comparison, and collection of literature such as journals, books, magazines and case studies, we were able to successfully identify the issues related to the Scrum method. Through further research, we were able to discover the relationship between the security principles and security in each of the Scrum phases. The second issue, then, was to enhance the Scrum model. The enhanced Scrum model that we have proposed has been evaluated in the requirement, development and testing phases. These research objectives were completed successfully. An agile team presented evaluations and feedback in regards to the enhancement model gathered from industry events. Team members who had experience using the agile model (team leaders, requirement engineers, developers and testers) evaluated whether agility was affected or not. Based on their evaluations, the research was considered relevant, and possible to implement. We also evaluated the degree of agility before and after enhancement using the 4-DAT framework.

The results showed that agility is improved if the security backlog (SB) is implemented, which means that agility is not negatively affected if the SB is added to the Scrum model. It is appropriate to clarify that, in this study, the agility of phases has not been considered. Thus, this is out of the scope of this study and can be considered in future research. In addition, the purpose of this study was not to prove that PB is unsuitable. On the contrary, the intention was to discover, after the enhancement of PB, how far Scrum agility was affected overall.

Acknowledgements

We wish to thank the teams at Twin System Company, Malaysia for their tremendous contribution in conducting the case study and Ministry of Science, Technology and Innovation (MOSTI), Malaysia to financially supporting this project under research grant vot 4S028.

References

[1] Dyba, T. and Dingsoyr, T., "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, Vol.50, pp.833-859, 2008. Article (CrossRef Link)

[2] Mchugh, O. and Conboy, k. and Lang, M., "Agile practices: The Impact on Trust in Software Project Teams," *IEEE Software*, Vol.29, pp.71-76, 2012. Article (CrossRef Link)

[3] Slaten, k., Droujkova, M., Berenson, S. B., Williams, L. and Layman, L., "Understanding Student Perceptions of Pair Programming and Agile Software Development Methodologies: Verifying a Model of Social Interaction," *IEEE Agile Conference*, pp.323-330, 2005. Article (CrossRef Link)

[4] Amir S. S., Amir A. S. and Fereidoon S., "Toward Empowering Extreme Programming from an Architectural Viewpoint," in *Proc. of 9th International Conference XP 2008*, Vol.9, pp.222-223, 2008. Article (CrossRef Link)

[5] Breivold, H. P., Sundmark, D., Wallin, P.and Larsson, S., "What Does Research Say about Agile and Architecture?," *IEEE Software Engineering Advances*, pp.32-37, 2010. Article (CrossRef Link)

[6] Wäyrynen, J., Bodén, M. and Boström, G., "Security engineering and eXtreme programming: an impossible marriage?," in *Proc. of 4th Conference on Extreme Programming and Agile Methods*, Vol.3134, pp.117-128, 2004. Article (CrossRef Link)

[7] Xiaocheng G., Richard F. P. and Fiona P., "Extreme Programming Security Practices", in *Proc. of 8th International Conference XP 2007*, Vol.4536, pp.226-230, 2007. Article (CrossRef Link)

[8] Sani, A., Firdaus, A., Jeong, S. R. and Ghani, I., "A Review on Software Development Security Engineering using Dynamic System Method(DSDM)," *International Journal of Computer Applications*, Vol.69, No.25, pp.33-44, 2013. Article (CrossRef Link)

[9] Ghani, I., Yasin, N. I. B., "Software Security Engineering in Extreme Programming Methodology: A Systematic Literature Review," *Journal Science International Lahore*, Vol.25, No.2, pp.215-221, 2013. Article (CrossRef Link)

[10] Firdaus, A., Ghani, I., Yasin, N. I. M., "Developing Websites using Feature Driven Development: A Case Study," *Journal of Clean Energy Technologies*, Vol.1, No.4, pp.322-326, 2013. Article (CrossRef Link)

[11] Sani, A., Ghani, I., Jeong, S. R., "Secure Dynamic System Development Method (Sdsdm) Model For Secure Software Development," *Journal, Science International Lahore, Special Issue*, 1059-64, 2013. Article (CrossRef Link)

[12] Sutherland, J. and Schwaber, K., "The Scrum Papers: Nut, Bolts, and Origin of an Agile Framework," *Scrum Inc*, 2011. Article (CrossRef Link)

[13] Julia, H. A., Sean, B., Robert, J., Ellison., Gary Mcgraw. And Nancy R.,"Software Security Engineering: A Guide for Project Manager," *Addison-Wesley Professional*, 2008. Article (CrossRef Link)

[14] Keramati, H. and Mirian-Hosseinabadi, S. H., "Integrating Software Development Security Activities with Agile Methodologies," in *Proc. of IEEE/ACS International Conference on Computer Systems and Applications*, pp.749-754, 2008. Article (CrossRef Link)
[15] Qumer, A. and Henderson-Sellers, B., "An evaluation of the degree of agility in six agile methods and it applicability for method engineering," *Information and Software Technology*, Vol.50, pp.280-295, 2008. Article (CrossRef Link)
[16] Walker, R., "Improving Software Economics: Top 10 Principles of Achieving Agility At Scale," *Improving Software Economics white paper*, 2009. Article (CrossRef Link)
[17] Lowell, L., Carmen, Z. and Erdogmus, H., "Extreme Programming and Agile Methods – XP/Agile Universe 2004," in *Proc. of 4th Conference on Extreme Programming and Agile Methods*, pp.121, 2004. Article (CrossRef Link)
[18] Erdogan, G., Meland, P. H. and Mathieson, D., "Security Testing in Agile Web Application Development – A Case Study Using the East Methodology," in *Proc. of 11th International Conference XP2010*, Vol.48, pp.14-27, 2010. Article (CrossRef Link)
[19] Brady, K., "AGILE/SCRUM Fails to get to grips with Human Psychology," at http://www.claretyconsulting.com/it/comments/agile-scrum-fails-to-get-to-grips-with-human-psychology.html, 2006. Article (CrossRef Link)
[20] Mikko, S., Richard, B. and Tapio, k., "Integrating Security into Agile Development Methods," in *Proc. of Proceedings of the 38th Hawaii International Conference on System Sciences*, 2005. Article (CrossRef Link)
[21] Anti, V, S., et al., "Secure software development and agile methods – notes," at http://confluence.agilefinland.com/display/af/Secure+software+development+and+agile+methods+-+notes, 2010. Article (CrossRef Link)
[22] Adrian, L., "FireStarter: Agile Development and Security," at https://securosis.com/blog/agile-development-and-security, 2010. Article (CrossRef Link)
[23] Zulkarnain Azham., Imran Ghani. and Norafida Ithnin., "Security Backlog in Scrum Security Practicesm," in *Proc. of 5th Malaysian Software Engineering Conference*, 2011. Article (CrossRef Link)
[24] McGraw, G., "Software Security: Building Security In," *Addison-wesley software security series*, 2006. Article (CrossRef Link)
[25] Checkland, P., "Soft Systems Methodology in Action," *Toronto, Ontario, Canada: John Wiley & Sons*, 1990. Article (CrossRef Link)
[26] QFD Institute, "Frequently Asked Questions About QFD," at http://www.qfdi.org/what_is_qfd/faqs_about_qfd.html, 2005. Article (CrossRef Link)
[27] Christel, M. and Kang, K., "Issues in Requirements Elicitation," *Software Engineering Institute*, 1992. Article (CrossRef Link)
[28] Kunz, Werner. and Rittel, Horst., "Issues as Elements of Information Systems," at http://www.cc.gatech.edu/~ellendo/rittel/rittel-issues.pdf, 1970. Article (CrossRef Link)
[29] Wood, J. and Silver, D., "Joint Application Design: How to Design Quality Systems in 40% Less Time," *New York: John Wiley & Sons*, 1989. Article (CrossRef Link)

[30] Schiffrin, D., “Approaches to Discourse,” *Oxford, UK: Blackwell*, 1994. Article (CrossRef Link)

[31] Hubbard, R., Mead, N. and Schroeder, C., “An Assessment of the Relative Efficiency of a Facilitator-Driven Requirements Collection Process with Respect to the Conventional Interview Method,” in *Proc. of 4th International Conference on Requirements Engineering*, pp.178-186, 2000. Article (CrossRef Link)

[32] Sullivan., “Security Development Lifecycle for Agile Development,” *Mirosoft*, at http://www.blackhat.com/presentations/bh-dc-10/Sullivan_Bryan/BlackHat-DC-2010-Sullivan-SDL-Agile-wp.pdf, 2009. Article (CrossRef Link)

[33] Gencer, E., "Security Testing of Web Based Applications," *Master Thesis Norwegian University of Science and Technology Department of Computer and Information Science*, 2009. Article (CrossRef Link)

[34] Sindre., G, and Opdahl. A., “Capturing security requirements through misuse cases,” in *Proc. of Proceedings of the 14th Norwegian informatics conference*, 2001. Article (CrossRef Link)

[35] Mullery., G, "CORE: A method for controlled requirements expression," in *Proc. of Proceedings of 4th International Conference on Software Engineering. (ICSE-4), pp.126 -135*, 1979. Article (CrossRef Link)

[36] Kang, C., Cohen, G., Hess, A., Novak, E., and Peterson, A., “Feature-oriented domain analysis (FODA) feasibility study,” *Technical report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University*, 1990. Article (CrossRef Link)

[37] Alnatheer, M., Nelson, K.: A proposed framework for understanding information security culture and practices in the Saudi context. In: Proceedings of the 7th Australian Information Security Management Conference, pp. 6–17. SECAU - Edith Cowan University, Australia, Perth, Australia , 2009. Article (CrossRef Link)

[38] Vaha-Sipila, A., “Software security in agile product management,” http://www.fokkusu.fi/agile-security/Software%20security%20in%20agile%20product%20management.pdf (2011) accessed on May 2013. Article (CrossRef Link)

Imran Ghani is a Senior Lecturer at Faculty of Computing, Universiti Teknologi Malaysia (UTM), Johor Campus. He received his Master of Information Technology Degree from UAAR (Pakistan), M.Sc. Computer Science from UTM (Malaysia) and Ph.D. from Kookmin University (South Korea). His research focus includes agile software development practices, semantics techniques, web services, software testing, enterprise architecture and software architecture.

Zulkarnain Azham earned his Bachelor's Degree in Software Engineering, Master's Degree in Computer Science (Information Security) from Universiti Teknologi Malaysia (UTM), Johor Campus. Currently, he is doing Ph.D. in Computer Science at Universiti Teknologi Malaysia (UTM). His research focus is on information security.

Seung Ryul Jeong is a Professor in the Graduate School of Business IT at Kookmin University, Korea. He holds a B.A. in Economics from Sogang University, Korea, an M.S. in MIS from University of Wisconsin, and a Ph.D. in MIS from the University of South Carolina, U.S.A. Dr. Jeong has published extensively in the information systems field, with over 60 publications in refereed journals like Journal of MIS, Communications of the ACM, Information and Management, Journal of Systems and Software, among others. Dr. Jeong's areas of interest are Process Management, Software Engineering, Systems Implementation, and Information Resource Management.