World of Computer Science and Information Technology Journal (WCSIT)
ISSN: 2221-0741
Vol. 3, No. 1, 15-19, 2013

S-Scrum: a Secure Methodology for Agile Development of Web Services

Davoud Mougouei, Nor Fazlida Mohd Sani, Mohammad Moein Almasi
Department of Information System
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia, 43400 Serdang,
Selangor, Malaysia

Abstract — Widely used in development of web services, Scrum contributes to agile service development, reducing the Time To Market (TTM) and increasing the profit to service providers. Caring for dynamic requirement changes and incremental development of web services are other advantages of employing Scrum for development of web services. However there are several problems with this methodology limiting its applicability to web service development. Scrum overlooks precise documentation of development activities to increase the development speed. Nonetheless this approach negatively affects the quality of the web services through incorporating imprecision and lack of tractability into the development process. On the other hand security as a quality attribute has always been one of the most important concerns of the web service development. To care for security of the web service we always need to incorporate security analysis and design into the development life cycle. Although there have been some attempts to care for analysis activities within the Scrum iterations, it is not clear yet how to achieve this automatically through the Scrum processes. On the other words careful engineering of security into the overall system analysis and design is often neglected. In this paper we propose a security-enhanced version of scrum i.e. Secure Scrum (S-Scrum) to accommodate security analysis and design activities within the Scrum. We have modified the scum process to care for security analysis and design through the standard Scum processes. The validity of the proposed approach is verified through formal modeling and description of the process steps. We propose a grammar for formal description of Scum process model.

Keywords-Scrum; Web Service; Security; Automata.

I. INTRODUCTION

The increasing demand for rapid development of web services has led to widely use of agile methodologies to facilitate rapid service identification and extraction of web services from legacy systems [1]. On the other hand security of web service has become a big concern of web service development. The importance of security is more tangible when it comes to significant [2] web services exposed globally over the web. Scrum as an agile methodology allows for rapid development of web services. Scrum also allows for keeping abreast of the changing requirements during the software development [3]. Evolutionary nature of Scrum process also facilitates incremental development of web services based on user requirements and market demand. However scrum is criticized for neglecting security analysis and design through the development processes [4]. In this regard, Scrum is not proper for secure software development [4] since it does not explicitly specify activities for security analysis and design required for secure software development [5]. To assure security of the web services it is required to present stuffiest documents proving satisfaction of security objectives [4]. Security assurance is not properly supported in Scrum due to the overlooking documentation of security considerations and analysis results. Neglecting the security documentation or de-emphasizing on documentation activates, is quiet risky in the context of developing security critical system [6]. Lack of security documentation also makes it difficult to adopt security standards such as Secure Systems Engineering Capability Maturity Model (SSE-CMM) [7] in scrum. SSE-CMM is designed to measure capabilities of the organization for developing secure software. Compared to SSE-CMM, Scrum does not care for essential security activities such as impact assessment, security risk assessment, threat identification and vulnerability analysis. Scrum process also does not support assurance building and security monitoring activities [4]. Formal and clear specification of web services as Security Critical Systems (SCSs) in the early stages of software development leads to more secure web services [5]. On the other hand tedious documentation activities required by traditional methodologies reduce the agility and compromises the benefits of Scrum. To support security assurance in Scrum a lightweight documentation is required to properly reflect the

security aspects of the system [2] while maintaining the benefits of Scrum. Such documentation must be produced through traceable analysis and design activities incorporated into the Scrum. Integrated into the Scrum, such a method must be simple, compatible with agile development and not intercepting the regular Scrum activities [3]. In order to be successfully integrated into the Scrum, such a method must specify all security considerations throughout the development activities. Although there have been some attempts to care for analysis activities within the Scrum iterations such as applying spikes [8], it is not yet clear how to achieve this automatically through the Scrum processes. Spikes are introduced to accommodate development activities which do not produce customer-valued products. However, it is not specified how to efficiently perform spiking and bundle the spikes to the scrum process activities.

In this paper we propose a security-enhanced version of Scrum i.e. Secure Scrum (S-Scrum) to incorporate security analysis and design activities into the Scrum processes. Our aim is to enable using Scrum for development of security critical web services. The proposed methodology addresses analysis and design of security needs within the early stages of software development throughout a constructive approach. We have modified the scrum processes to accommodate documentation actives which reflect the security aspects of the target web service. Our proposed methodology properly cares for both security and also changing requirements during the release planning, sprints and spikes. Spikes are properly employed in S-Scrum to accommodate analysis, design and detailed design activities for security requirements. We illustrate the process steps through formal modeling of the S-Scrum. Non-deterministic finite automatons are used to demonstrate the process steps and transition among them. We also have proposed a context free grammar for formal description of process steps. The remainder of the paper is organized as follows. Section II presents our proposed methodology. In Section III we bring the formal model of the proposed methodology and describe the proposed grammar. Section IV of the paper shows the validity of our approach through applying it to a running example. Paper will be concluded in section V with some general remarks.

II. THE PROPOSED METHODOLOGY

The proposed methodology confirms to the standard scrum process as depicted in Figure 1. The process starts with initiating the product backlog. Then the process continues in two simultaneous branches. The first process starts story gathering to care for change in requirements made by stakeholders. The incoming changes will be kept in the waiting list till the start of next spring unless the change has a crucial effect on the implementation of the web service. In this case we terminate the current release process and start again from release planning step.

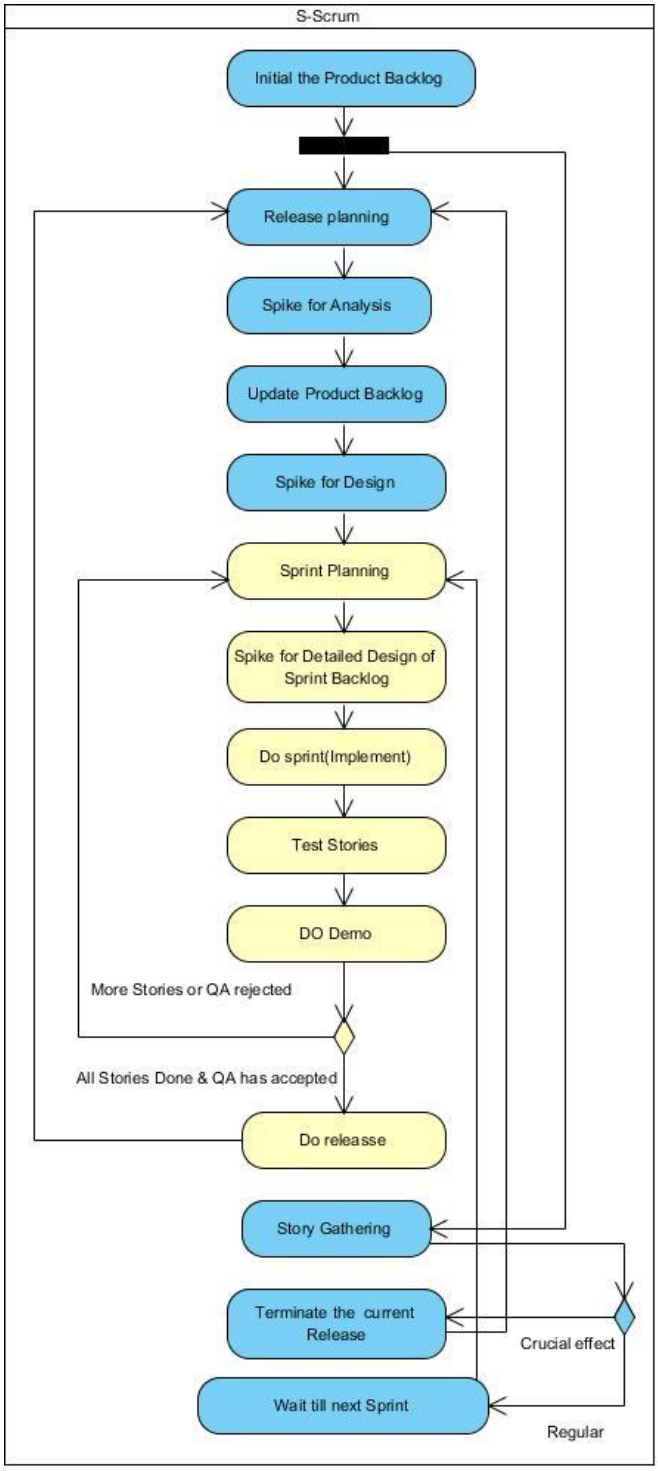

Figure 1. S-Scrum process

After release planning spike for security analysis will be performed and after that product backlog may be updated. The process then continues with performing security modeling in next spike. In this stage the results of the security analysis will be reflected into the model of current release product items. The high priority product items will be chosen then during the sprint planning to be implemented during the sprint. However

we still need to incorporate details of security analysis into the model of these selected items within the detailed design spike. The process will be followed by implementing, testing and demo [8] of the current sprint's product items. As standard scrum practice, if the stories are complemented and accepted by quality assurance unit, the release will be conducted. Otherwise more sprints will be required to resolve the issues. The process continues till to the point all stories are implemented. Since our focus in this work is more to incorporate the security considerations into the scrum process, a penetration test or similar security testing may be conducted within the test stage. In this work we are not going to propose security analysis techniques for security analysis and design stages. However all possible security artifacts including misuse case [9] diagrams may be used for this purpose. All the process steps are depicted in Figure 1.

III. FORMAL MODEL OF THE PROPOSED METHODOLOGY

To facilitate further analysis of the development process, we present formal representation of S-Scrum. For this purpose, we model the process using non-deterministic finite automatons and using context free grammars. Figure 2 demonstrates the basic automaton model of the S-Scrum. Table 1 describes the labels used in Figure 2.

TABLE I. STATES IN S-SCRUM AYTOMATON

State/Non terminal	Description
A	*Spike 1(analysis)*
B	*Spike 2(modeling)*
C	*Sprint*
D	*Last Sprint*
E	*Release*
F	*New Requirement*
G	*Backlog Update*

As depicted in Figure 2, we have formally modeled the S-Scrum process starting with the first spike i.e. security analysis spike. The first spike as given in TABLE I is represented as state A which is the initial state of the automata in Figure 2. The proposed grammar for description of S-Scrum process is formulated in TABLE III. process starts with initiating the product backlog. Then the process continues in two simultaneous branches. The first process starts story gathering to care for change in requirements made by stakeholders. The incoming changes will be kept in the waiting list till the start of next spring unless the change has a crucial effect on the implementation of the web service. In this case we terminate the current release process and start again from release planning step. After release planning spike for security analysis will be performed and after that product backlog may be updated. The process then continues with performing security modeling in next spike. In

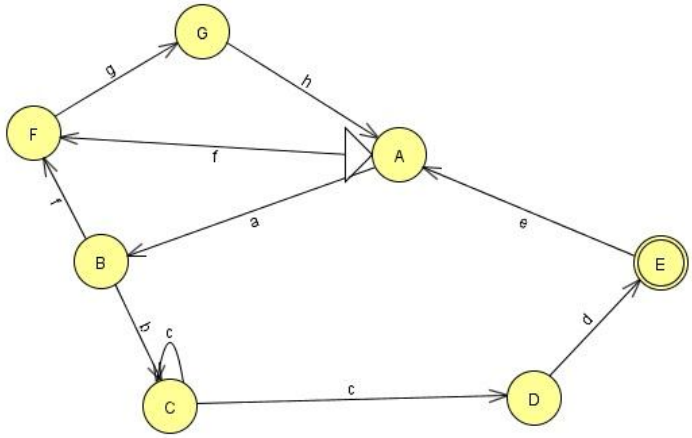

Figure 2. Automata model for S-Scrum

TABLE II. TRANSITION LABELES IN S-SCRUM AYTOMATON

Terminals	Description
a	*Security analysis done*
b	*Security modeling done*
c	*Stories implemented*
d	*Finalized*
e	*New release*
f	*New requirement*
g	*New requirement Prioritized*
h	*Backlog updated*

In this stage the results of the security analysis will be reflected into the model of current release product items. The high priority product items will be chosen during the sprint planning to be implemented during the sprint. However, we still need to incorporate details of security analysis into the model of these selected items within the detailed design spike. The process will be followed by implementing, testing and demo [8] of the current sprint's product items. As standard scrum practice, if the stories are complemented and accepted by quality assurance unit, the release will be conducted otherwise more sprints will be required to resolve the issues. The process continues till to the point all stories are implemented. Since our focus in this work is more to incorporate the security considerations into the scrum process, a penetration test or similar security testing may be conducted within the test stage. In this work we are not going to propose security analysis techniques for security analysis and design stages. However all possible security artifacts including misuse case [9] diagrams may be used for this purpose. All the process steps are depicted in Figure 1.

The production rules for the proposed grammar also are given in Table III.

TABLE III. PRODUCTION RULES

Rules
$P1: A \rightarrow aBD \mid fF$
$P2: B \rightarrow bC \mid fF$
$P3: C \rightarrow Cc \mid \lambda$
$P4: D \rightarrow dE$
$P5: E \rightarrow eA \mid \lambda$
$P6: F \rightarrow gG$
$P7: G \rightarrow hA$

IV. RUNNING EXCAMPLE

We have demonstrated the validity of our approach by applying it to a basic scenario for implementation of security requirements in a money transfer web service. The scenario is as follows. The development team aims to develop security requirements of a Money Transfer Service (MTS) using scrum as the development methodology. MTS is described as follows:

OBS provides some standard banking services including money transfer service over the internet. The bank accounts are a tempting target for hackers. For this reason, MTS transactions must be protected to keep financial losses to a minimum. The availability of MTS is as important as the confidentiality and integrity. The MTS also has a server which should be protected from any possible misuse. In addition to that, an attacker may exploit the MTS' internal communication network to threaten the transactions.

MTS in addition should prevent unauthorized online access to the service. Thus, it supports user authentication by checking the user name and password. However, the attacker still can guess either user name or password but it is supposed to be difficult. MTS must offer reasonable assurance that their customers' accounts are secure. The main threat that concerns MTS is that an attacker will transfer money out of customers' accounts.

The result of security analysis for MTS based on [10] is demonstrated in Figure 3. During the scrum process the development team implements the requirements R1.1.3.7.1 and R1.1.3.7.2 during the first release. The two security requirements R.1.1.3.8.1 and R1.1.3.8.2 will be asked by stakeholders to be implemented during the second release of the product. The activities required to implement the security requirements for MTS are listed in Table IV.

TABLE IV. PROCESS ACTVITIES FOR MTSSECURITY IMPLEMENTATION

Terminals	Description
a	*Performing security analysis for MTS*
b	*Conducting security modeling of MTS*
c	*Implementation of security requirement of MTS*
d	*First release of MTS*
e	*New release for MTS*
f	*Catering for the new requirements of 1. Form field manipulation 2. Query string manipulation*
g	*Prioritization of requirements take place*
h	*Proceed to update backlog*
a	*Analyze new requirements*
b	*Design new requirements*
c	*Implementation of new requirements*
d	*Second release of MTS includes new requirements*

V. CONCLUSION

This paper has proposed a security-enhanced version of Scrum i.e. Secure Scrum (S-Scrum) to incorporate security analysis and design activities into the Scrum processes. The proposed methodology has modified the scrum processes to accommodate documentation actives which reflect the security aspects of the target web service. Our proposed methodology cares for both security and also changing requirements during the release planning, sprints and spikes. We illustrate the process steps through formal modeling of the S-Scrum. Non-deterministic finite automatons are used to demonstrate the process steps and transition among them. We also have proposed a context free grammar for formal description of process steps. The proposed methodology is partially modeled and applied on a typical scenario for development of a money transfer service to show the validity of our approach.

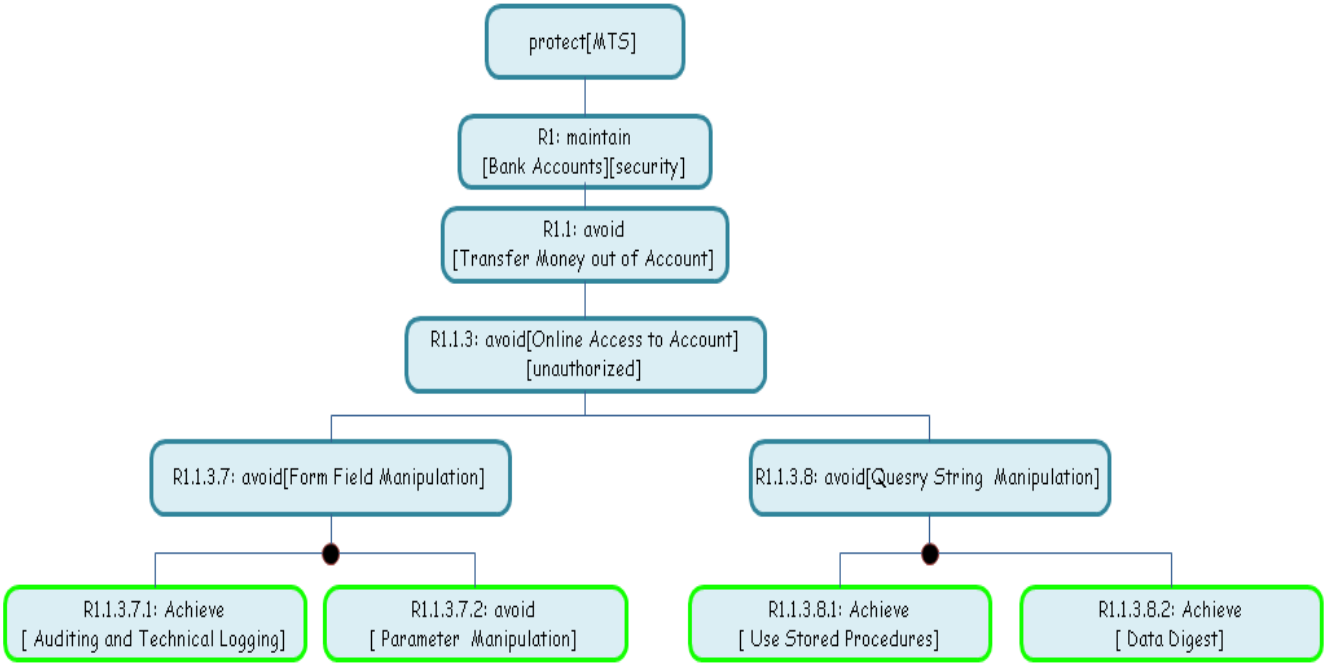

Figure 3. MTS' security requirement model for form field manipulation and query string manipulation

REFERENCES

[1] S. Chung, D. H. Won, S.-H. Baeg, and S. Park, "A Model-Driven Scrum Process for Service-Oriented Software Reengineering: mScrum4SOSR," in Computer Science and its Applications, 2009. CSA '09.2nd International Conference on, ed,2009, pp.1-8.

[2] H. Chivers, R. Paige, and X. Ge, "Agile Security Using an Incremental Security Architecture," in Extreme Programming and Agile Processes in Software Engineering vol. 3556, H. Baumeister, M. Marchesi, and M. Holcombe, Eds., ed: Springer Berlin / Heidelberg, 2005, pp. 1325-1327.

[3] M. U. Kumar, D. S. Kumar, B. P. Rani, K. V. Rao, A. V. K. Prasad, and D. Shravani, "Dependable Solutions Design by Agile Modeled Layered Security Architectures," in Advances in Computer Science and Information Technology. Networks and Communications vol. 84, N. Meghanathan, N. Chaki, D. Nagamalai, O. Akan, P. Bellavista, J. Cao, F. Dressler, D. Ferrari, M. Gerla, H. Kobayashi, S. Palazzo, S. Sahni, X. S. Shen, M. Stan, J. Xiaohua, A. Zomaya, and G. Coulson, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 510-519.

[4] J. Wäyrynen, M. Bodén, and G. Boström, "Security Engineering and eXtreme Programming: An Impossible Marriage?," in Extreme Programming and Agile Methods - XP/Agile Universe 2004 vol. 3134, C. Zannier, H. Erdogmus, and L. Lindstrom, Eds., ed: Springer Berlin / Heidelberg, 2004, pp. 152-195.

[5] J. Viega and G. McGraw, Building secure software: how to avoid security problems the right way: Addison-Wesley, 2002.

[6] B. Boehm, "Get Ready for Agile Methods, with Care," Computer, vol. 35, pp. 64-69, 2002.

[7] F. Jos, B. Nunes, A. D. Belchior, and A. B. Albuquerque, "Security Engineering Approach to Support Software Security," presented at the Proceedings of the 2010 6th World Congress on Services, 2010.

[8] S. Resnick, A. Bjork, and M. de la Maza, "Professional Scrum with Team Foundation Server 2010," 2011.

[9] T. Lodderstedt, D. A. Basin, and J. u. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," in Proceedings of the 5th International Conference on The Unified Modeling Language, ed. London, UK, UK: Springer-Verlag, pp. 426-441.

[10] M. M. D. Mougouei, S. Moradmand, "A Goal-Based Modeling Approach to Develop Security Requirements of Fault Tolerant Security-Critical Systems," in 4th International Conference on Computer and Communication Engineering, ed. Malaysia, 2012.